

# Outsourced Cloud Data Privacy-Preserving Framework: An Efficient Broadcast Encrypted Search Realization

Yibo Cao<sup>a</sup>, Shiyuan Xu<sup>b</sup>, Xiu-Bo Chen<sup>a</sup>, Gang Xu<sup>c,a</sup>, Siu-Ming Yiu<sup>b</sup>,  
Zongpeng Li<sup>d</sup>

<sup>a</sup>*Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China*

<sup>b</sup>*Department of Computer Science, School of Computing and Data Science, The University of Hong Kong, Pok Fu Lam, 999077, Hong Kong,*

<sup>c</sup>*School of Artificial Intelligence and Computer Science, North China University of Technology, Beijing, 100144, China*

<sup>d</sup>*Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, 100084, China*

---

## Abstract

The development of cloud networks facilitates data outsourcing, sharing, and storage, but it has also raised several security concerns. Public key authenticated encryption with keyword search (PAEKS) enables the encrypted search over cloud data while resisting the insider keyword guessing attacks (IKGAs). However, existing PAEKS schemes are limited to a single receiver, restricting application prospects in cloud networks. In addition, quantum computing attacks and key leakage issues further threaten data security, which has attracted extensive attention from researchers. Therefore, designing an encrypted search scheme to resist the above-mentioned attacks is still far-reaching. In this paper, we first propose BroSearch, an outsourced data privacy-preserving framework through efficient broadcast encrypted search for cloud networks. It utilizes lattice sampling algorithms to authenticate the keyword and offers searchability over broadcasting ciphertext while enjoying IKGAs-resistance in a quantum setting. To get around key leakage issues, we then incorporate the minimal cover set technique and lattice basis extension algorithm to construct FS-BroSearch as an enhanced version. Furthermore, we give a rigorous security analysis and a comprehensive performance evaluation of BroSearch and FS-BroSearch. Specifically, BroSearch consumes only 61.11%, 81.82%, and 83.33% of the execution time compared to prior art in terms of ciphertext calculation, trapdoor generation, and search procedures,

which is practical and efficient in cloud networks.

*Keywords:* Cloud networks, privacy-preserving, encrypted search, ciphertext broadcast, keyword authentication, forward security

---

## 1. Introduction

Cloud computing has revolutionized data outsourcing by enabling businesses to store and process large volumes of data remotely, offering significant advantages in terms of cost-efficiency and accessibility [1], [2]. With cloud networks, users can access their data from anywhere, eliminating the need for physical infrastructure and reducing local storage overhead [3]. However, this convenience raises serious privacy concerns [4], [5]. Many researchers have turned to public key encryption with keyword search (PEKS) techniques, enabling secure encrypted search for cloud data sharing through cloud networks [6], [7], [8]. Despite these advancements, the cloud server, as a semi-honest entity, is vulnerable to a variety of malicious attacks, including internal keyword guessing attacks (IKGAs) [9], [10]. To alleviate this issue, public key authenticated encryption with keyword search (PAEKS) [11], [12], [13] has been widely studied. It involves encrypting keywords while authenticating them using a secret key of the data sender to resist IKGAs, thereby further achieving the privacy-preserving of cloud data.

Most PAEKS primitives are primarily designed for a single-receiver model [11], [14], [15], [16], [17], [18], which involves four key entities: the data sender, the data receiver, the trusted authority, and the cloud server, as illustrated in Fig. 1. However, the expanding scale of cloud networks exposes significant limitations for this model. In most cases, the same data is distributed to multiple receivers simultaneously [19]. For instance, in a healthcare cloud service, the encrypted medical data owned by a specific patient is uploaded to the cloud, and multiple healthcare professionals (e.g., doctors, nurses, and pharmacists) need simultaneous access to collaborate effectively [20], [21]. A straightforward approach to mitigating this issue is to perform point-to-point encryption repeatedly. Nonetheless, the computational overhead grows significantly as the number of receivers increases. A more efficient alternative is broadcast authenticated encryption with keyword search (BAEKS), which has been formalized in previous studies [22], [23], [24]. BAEKS can broadcast a keyword ciphertext on a public channel, allowing a set of receivers to search it without incurring the additional computational overhead associated with

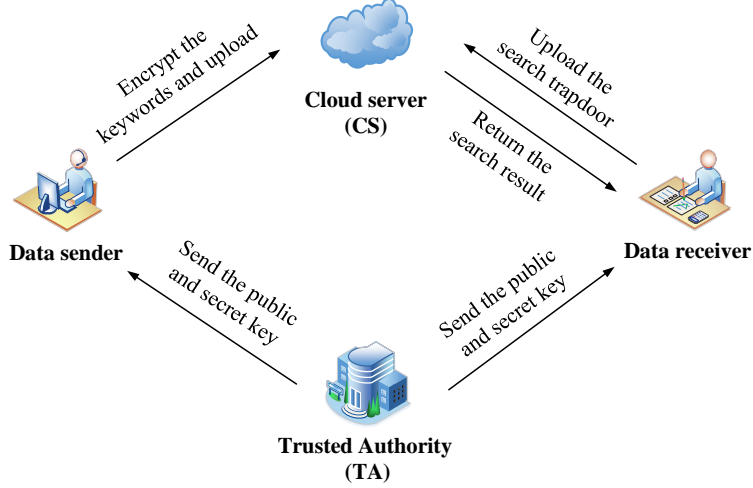


Figure 1: A single-receiver encrypted search model in cloud networks.

multiple point-to-point encryptions.

Unfortunately, existing BAEKS schemes still face two challenges. On the one hand, these schemes rely on classical hardness assumptions (e.g., the discrete logarithm hardness), which are vulnerable to quantum computing attacks [25]. Lattice-based cryptography, a quantum-resistant primitive widely adopted by researchers for data privacy-preserving in cloud networks [14], [15], [26], [27], [28], [29], offers a promising solution to this issue. On the other hand, in practical cloud applications, a malicious adversary could calculate a trapdoor corresponding to a specific keyword if it gains access to a data receiver’s secret key [30], [16]. Then, the adversary sends it to the cloud server, allowing it to match the keyword ciphertext and thereby significantly compromise keyword security. To address this concern, numerous researchers have introduced the concept of forward security to several cryptographic systems [31], [32], [33], but there does not exist a forward secure BAEKS scheme as so far.

Given this, IKGAs, quantum computing attacks, and key leakage issues are three serious threats that need to be addressed urgently in cloud networks. This situation leads us to the main questions addressed in this work:

*Can we design an outsourced cloud data privacy-preserving framework to resist the above-mentioned threats?*

In this paper, we address the aforementioned question in two milestones.

Firstly, we propose BroSearch, which can broadcast the keyword ciphertext to a set of receivers while preserving the functionality of encrypted search. Additionally, it offers enhanced data privacy-preserving in complex cloud environments, as it is resistant to both IKGAs and quantum computing attacks. To resist IKGAs, we embed a public matrix  $\mathbf{U}$  into the ciphertext and trapdoor, while invoking the **SampleLeft** algorithm to achieve keyword authentication. However, existing lattice-based PAEKS schemes [15], [34], [35] do not support ciphertext broadcast capability. Addressing these issues is not trivial. Unlike the above-mentioned schemes, we embed the public keys of all receivers in a broadcast list  $\mathcal{L}$  into the ciphertext. Furthermore, during the trapdoor generation process, we innovatively leverage the **GenSamplePre** algorithm, allowing the calculation of a valid trapdoor by using the secret key of any receiver in  $\mathcal{L}$ . As a result, BroSearch successfully facilitates both ciphertext broadcast and encrypted search for cloud data.

Secondly, we extend BroSearch to develop the FS-BroSearch in response to the key leakage issue. Inspired by Yu et al.[31], we introduce the binary tree structure, the minimal cover set technique, and a lattice basis extension algorithm to update the secret key. Thus, even if the secret key of a specific receiver is compromised during a given time period, an adversary is unable to generate a valid search trapdoor for any previous time periods, thus FS-BroSearch can mitigate the secret key leakage.

In a nutshell, our contributions are summarized as follows:

- We present BroSearch, an outsourced data privacy-preserving framework through efficient broadcast encrypted search for cloud networks. BroSearch enables the simultaneous broadcasting of keyword ciphertext to a set of receivers while providing resilience against IKGAs and quantum computing attacks. Additionally, we propose an enhanced version of BroSearch, named FS-BroSearch, which mitigates the key leakage issue.
- We construct BroSearch by leveraging lattice algebraic structures and lattice sampling algorithms. Specifically, by invoking the **GenSamplePre** algorithm initially, each receiver in the broadcast set can generate a valid search trapdoor. Furthermore, in FS-BroSearch, we introduce a binary tree structure, the minimal cover set technique, and a lattice basis extension algorithm to enable time-period representation and facilitate the secret key update for data receivers.

- We provide the IND-CKA and IND-IKGA security models for BroSearch and FS-BroSearch, and show a rigorous security analysis to demonstrate that their security can be reduced to the Learning With Errors (LWE) hardness, thereby ensuring their post-quantum security.
- We offer a detailed performance evaluation of BroSearch and FS-BroSearch in terms of both computational and communication overhead. In particular, the **Encrypt**, **trapdoor**, and **Search** algorithms in BroSearch consume 61.11%, 81.82%, and 83.33% of the time required by other state-of-the-art schemes, which is efficient in cloud networks.

The remainder of this paper is structured as follows. Section 2 presents numerous related works to showcase recent advancements. Following that, Section 3 provides an introduction to the preliminary concepts. The system model, formal definitions, and security models for BroSearch and FS-BroSearch are then depicted in Section 4. A detailed explanation and its security analysis of BroSearch are demonstrated in Section 5, while Section 6 focuses on the FS-BroSearch. In Section 7, we delve into the performance evaluation and comparison. Finally, we summarize this paper in Section 8.

## 2. Related Works

Huang et al. introduced a public-key authenticated encryption with keyword search (PAEKS) scheme to implement keyword authentication through a data owner’s secret key [11]. To achieve scalability for healthcare cloud storage, Cheng et al. presented a server-aided PAEKS scheme, ensuring the size of the ciphertext and trapdoor constant [18]. Chen et al. proposed a public-key authentication encryption with similar data search for pay-per-query, namely PAESS, which can prevent cloud servers and data users from colluding to deny chargebacks [36]. Liu et al. put forward a generic construction for PAEKS and an instantiation over lattice to achieve the anti-quantum property [14], and enhanced its security [37]. Furthermore, Cheng et al. pointed out some security issues [37], [24], and constructed two PAEKS schemes over lattice [15]. Following that, Luo et al. integrated attribute-based encryption [34] and proxy re-encryption [35] into lattice-based PAEKS to enhance the practicality for cloud systems.

Since encrypted messages can be decrypted by a specified group of authorized users, broadcast encryption (BE), first introduced by Fiat et al. [38], is often considered more practical in cloud networks. To enhance anonymity,

Bae et al. combined a message authentication scheme with beacon encryption, and then put forward an inter-vehicle broadcast authentication with encryption scheme for vehicle-to-vehicle (V2V) communication [39]. After that, Zhang et al. provided an identity-based broadcast proxy re-encryption scheme for fully anonymous data sharing [40]. Yin et al. constructed a new dual-mode identity-based BE scheme, named DM-IBBE, to protect sensitive data in smart contracts [41].

Ali et al. foresaw the combinability of BE and PEKS, and constructed a broadcast SE scheme, which is a novel cryptographic primitive to search the keyword ciphertext encrypted by the public key of a group of specified data users [42]. Enlightened by the concept of PAEKS, Liu et al. constructed the BAEKS cryptographic primitive to resist IKGAs, and the ciphertext and trapdoor security were proved under the DBDH assumption [22]. Mukherjee introduced a stronger security model and ensured the ciphertext and trapdoor security in the standard model [23]. Emura et al. put forward a generic construction of fully anonymous BAEKS, which provides the anonymity and consistency of keyword ciphertext and supports a multi-receiver model [24]. However, none of the aforementioned schemes can resist quantum computing attacks, and no post-quantum encrypted search scheme supporting ciphertext broadcast so far.

In 2019, a lattice-based forward secure public key with keyword search (FS-PEKS) scheme was proposed by Zhang et al., which utilized lattice basis delegation to update the secret key [43]. After that, Yu et al. introduced the binary tree structure, minimal cover set technique, and lattice basis extension to construct an efficient FS-PEKS scheme over lattice [31]. For PAEKS primitive, Xu et al. constructed a forward secure PAEKS over lattice, namely FS-PAEKS, to achieve the IND-CKA and IND-IKGA secure [16].

To sum up, there exists a valuable requirement to construct an encrypted search scheme with ciphertext broadcast and extend it under the forward security property for resisting IKGAs, quantum computing attacks, and key leakage issues.

### 3. Preliminaries

**Definition 1.** Suppose a matrix  $\mathbf{M} = (\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_m)$  is composed of  $m$  linearly independent vectors, the lattice  $\Lambda$  is defined as:  $\Lambda = \Lambda(\mathbf{M}) = \{x_1\mathbf{m}_1 + x_2\mathbf{m}_2 + \dots + x_m\mathbf{m}_m | x_i \in \mathbb{Z}, i \in [m]\}$ , where  $\mathbf{M}$  is a lattice basis of  $\Lambda$ .

**Definition 2.** Suppose three integers  $n, m, q$ , and a matrix  $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$ , a  $q$ -ary integer lattice is defined as:

$$\Lambda_q(\mathbf{M}) := \{\mathbf{v} \in \mathbb{Z}^m \mid \exists \mathbf{s} \in \mathbb{Z}_q^n, \mathbf{M}^\top \mathbf{s} = \mathbf{v} \bmod q\}.$$

$$\Lambda_q^\perp(\mathbf{M}) := \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{M}\mathbf{v} = \mathbf{0} \bmod q\}.$$

$$\Lambda_q^{\mathbf{u}}(\mathbf{M}) := \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{M}\mathbf{v} = \mathbf{u} \bmod q\}.$$

**Definition 3.** Suppose a parameter  $\sigma \in \mathbb{R}^+$ , a center  $\mathbf{c} \in \mathbb{Z}^m$ , and any vector  $\mathbf{v} \in \mathbb{Z}^m$ , the discrete Gaussian distribution over  $\Lambda$  is defined as:  $\mathcal{D}_{\Lambda, \sigma, \mathbf{c}}(\mathbf{v}) = \frac{\rho_{\sigma, \mathbf{c}}(\mathbf{v})}{\rho_{\sigma, \mathbf{c}}(\Lambda)}$ , for  $\forall \mathbf{v} \in \Lambda$ , where  $\rho_{\sigma, \mathbf{c}}(\mathbf{v}) = \exp(-\pi \frac{\|\mathbf{v} - \mathbf{c}\|^2}{\sigma^2})$  and  $\rho_{\sigma, \mathbf{c}}(\Lambda) = \sum_{\mathbf{v} \in \Lambda} \rho_{\sigma, \mathbf{c}}(\mathbf{v})$ .

**Definition 4.** Suppose  $m$  independent pairs  $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ , and each sample is governed by the following either one to define the decisional  $\text{LWE}_{n, m, q, \chi}$  assumption:

1. Pseudo-random sample:  $(\mathbf{a}_i, b_i) = (\mathbf{a}_i, \mathbf{a}_i^\top \mathbf{s} + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ , where  $\mathbf{s}$  is a randomly vector,  $e_i$  is an error, and  $\mathbf{a}_i$  is an uniform vector.
2. Random sample: Randomly samples from  $\mathbb{Z}_q^n \times \mathbb{Z}_q$ .

Besides, the decisional  $\text{LWE}_{n, m, q, \chi}$  assumption is as hard as the worst-case SIVP and GapSVP problem [44].

**Lemma 1.** [45] Suppose two positive integers  $n, m$ , a prime  $q$ , where  $m = \Theta(n \log q)$ , the TrapGen algorithm returns a full-rank matrix  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$  and its basis  $\mathbf{T}_{\mathbf{A}} \in \mathbb{Z}^{m \times m}$  over  $\Lambda_q^\perp(\mathbf{A})$ , such that  $\mathbf{A}$  is  $\text{negl}(n)$ -close to uniform and  $\|\widetilde{\mathbf{T}}_{\mathbf{A}}\| = O(\sqrt{n \log q})$  with all but negligible probability in  $n$ .

**Lemma 2.** [46] Suppose four integers  $n, m, m_1, q$ , and two matrices  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{M}_1 \in \mathbb{Z}_q^{n \times m_1}, \mathbf{T}_{\mathbf{A}} \in \mathbb{Z}_q^{m \times m}$  is a basis of  $\Lambda_q^\perp(\mathbf{A})$ ,  $\mathbf{u} \in \mathbb{Z}_q^n$  is a vector, and  $\sigma > \|\widetilde{\mathbf{T}}_{\mathbf{A}}\| \cdot \omega(\sqrt{\log(m + m_1)})$ , the SampleLeft algorithm returns a vector  $\mathbf{e} \in \mathbb{Z}_q^{m+m_1}$ , such that  $(\mathbf{A} \mid \mathbf{M}_1)\mathbf{e} = \mathbf{u} \bmod q$ .

**Lemma 3.** [46] Suppose four integers  $n, k, m, q$ , and three matrices  $\mathbf{A} \in \mathbb{Z}_q^{n \times k}, \mathbf{B} \in \mathbb{Z}_q^{n \times m}, \mathbf{R} \in \mathbb{Z}^{k \times m}, \mathbf{T}_{\mathbf{B}} \in \mathbb{Z}_q^{m \times m}$  is a basis of  $\Lambda^\perp(\mathbf{B})$ ,  $\mathbf{u} \in \mathbb{Z}_q^n$  is a vector, and  $\sigma > \|\widetilde{\mathbf{T}}_{\mathbf{B}}\| \cdot \|\mathbf{R}\| \omega(\sqrt{\log(m)})$ , the SampleRight algorithm returns a vector  $\mathbf{e} \in \mathbb{Z}_q^{m+k}$ , such that  $(\mathbf{A} \mid \mathbf{A}\mathbf{R} + \mathbf{B})\mathbf{e} = \mathbf{u} \bmod q$ .

Suppose four positive integers  $n, m, q, k$ , a matrix  $\mathbf{A} = (\mathbf{A}_1 \mid \cdots \mid \mathbf{A}_k) \in \mathbb{Z}_q^{n \times km}$ , and a set  $\mathcal{M} = \{i_1, i_2, \dots, i_j\} \subset [k]$ , we set  $\mathbf{A}_{\mathcal{M}} := (\mathbf{A}_{i_1} \mid \mathbf{A}_{i_2} \mid \cdots \mid \mathbf{A}_{i_j}) \in \mathbb{Z}_q^{n \times jm}$ . Then, we introduce the Lemma 4 as follows:

**Lemma 4.** [47] Suppose four positive integers  $n, m, q, k$ , where  $q \geq 2$ , and  $m \geq 2n \log q$ . After input a matrix  $\mathbf{A} \in \mathbb{Z}_q^{n \times km}$ , a lattice basis  $\mathbf{T}_{\mathbf{A}_{\mathcal{M}}}$  for  $\Lambda_q^\perp(\mathbf{A}_{\mathcal{M}})$ , a set  $\mathcal{M} \subset [k]$ , a vector  $\mathbf{u} \in \mathbb{Z}_q^n$ , and a Gaussian parameter  $\sigma \geq \|\widetilde{\mathbf{T}_{\mathbf{A}_{\mathcal{M}}}}\| \cdot \omega(\sqrt{\log km})$ , the PPT algorithm  $\text{GenSamplePre}(\mathbf{A}, \mathbf{T}_{\mathbf{A}_{\mathcal{M}}}, \mathcal{M}, \mathbf{u}, \sigma)$  will output a vector  $\mathbf{e} \in \mathbb{Z}^{km}$  statistically close in  $\mathcal{D}_{\Lambda_q^u(\mathbf{A}), \sigma}$ , such that  $\mathbf{A}\mathbf{e} = \mathbf{u} \bmod q$ .

**Lemma 5.** [47] Suppose four positive integers  $n, m, m', q$ , two matrices  $\mathbf{A} \in \mathbb{Z}^{n \times m}$ ,  $\mathbf{A}' \in \mathbb{Z}^{n \times m'}$ . After input  $\mathbf{A}'' = (\mathbf{A} \mid \mathbf{A}') \in \mathbb{Z}_q^{n \times (m+m')}$ , and a basis  $\mathbf{T}_{\mathbf{A}} \in \mathbb{Z}_q^{m \times m}$  for  $\Lambda_q^\perp(\mathbf{A})$ , the deterministic polynomial time (DPT) algorithm  $\text{ExtBasis}(\mathbf{A}'', \mathbf{S})$  will calculate a lattice basis  $\mathbf{T}_{\mathbf{A}''}$  for  $\Lambda_q^\perp(\mathbf{A}'') \subseteq \mathbb{Z}_q^{m \times m''}$ , where  $\|\widetilde{\mathbf{T}_{\mathbf{A}}}\| = \|\widetilde{\mathbf{T}_{\mathbf{A}''}}\|$ ,  $m'' = m + m'$ .

**Lemma 6.** [46] Suppose a lattice  $\Lambda(\mathbf{A})$  and its lattice basis  $\mathbf{T}_{\mathbf{A}}$ , we obtain:  $\Pr[\|\mathbf{v}\| > \sigma\sqrt{m} : \mathbf{v} \leftarrow \mathcal{D}_{\Lambda_q^u(\mathbf{A}), \sigma}] \leq \text{negl}(m)$ , where  $\sigma \geq \|\widetilde{\mathbf{T}_{\mathbf{A}}}\| \cdot \omega(\sqrt{\log m})$ , and  $\text{negl}(\cdot)$  is a negligible function.

**Lemma 7.** [48] Suppose four positive integers  $n, m, m', k, q, \sigma$ , two matrices  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ ,  $\mathbf{U} \in \mathbb{Z}_q^{n \times k}$ , and a matrix  $\mathbf{R} \in \mathbb{Z}^{m \times k}$  sampled from the distribution  $\mathcal{D}_\sigma(\Lambda_q^u(\mathbf{A}))$  and a matrix  $\mathbf{S}$  uniformly selected from  $\{-1, 1\}^{m \times m'}$ , then the followings hold:  $\|\mathbf{R}^\top\|_2 \leq \sigma\sqrt{mk}$ ,  $\|\mathbf{R}\|_2 \leq \sigma\sqrt{mk}$ , and  $\|\mathbf{S}\|_2 \leq 20\sqrt{m}$ .

## 4. Framework Description

We provide our outsourced cloud data privacy-preserving framework and describe the formal definitions and security models of BroSearch and FS-BroSearch. Table 1 clarifies the acronyms and descriptions.

### 4.1. System Models

The system model in our paper is illustrated in Fig. 2, which contains four participating entities: trusted authority, data sender, data receivers, and cloud server.



Table 1: Glossary

Acronym	Definition
$l$	the number of data receivers
$\tau$	the level number of the binary tree
$T$	the number of time period, where $T = 2^\tau$
$\mathcal{L}$	the broadcast list
<b>ck</b>	the keyword owned by data sender
<b>tk</b>	the keyword to be searched by data receiver
$(\mathbf{pk}_S, \mathbf{sk}_S)$	the public and secret keys of data sender
$(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i})$	the public and secret keys of data receiver $i$ , where $i \in \mathcal{L}$
$\text{CT}(\text{resp. CT}_t)$	the ciphertext (resp. with time period $t$ )
$\text{TD}(\text{resp. TD}_t)$	the trapdoor (resp. with time period $t$ )

1. **Trusted authority (TA)**: TA is charged with initializing the system to obtain the public parameters and calculate the public and secret keys for the data sender and receivers. To facilitate trapdoor generation, the public keys of all receivers in the broadcast list are packaged and sent to each receiver. TA is a *fully trusted* entity in this system.
2. **Data sender**: As a *fully trusted* entity, the data sender can extract the keywords from data and calculate a valid ciphertext to the cloud server with its secret key and several public keys of the receivers in a broadcast list.
3. **Data receivers**: Since the data receiver's secret key may be stolen by a malicious adversary, it is considered *semi-honest*. When a data receiver in the broadcast list has a search requirement, it generates a trapdoor to the cloud server by its secret key, and then receives a search result.
4. **Cloud server (CS)**: CS is *semi-honest*, upon receiving a keyword ciphertext from the data sender and a search trapdoor from the data receiver, CS executes the search procedure to return a search result to the receiver.

#### 4.2. Formal Definitions

Our BroSearch is defined as  $\Pi_{\text{BroSearch}} = (\text{Setup}, \text{KeyGen}_S, \text{KeyGen}_R, \text{Encrypt}, \text{Trapdoor}, \text{Search})$ .

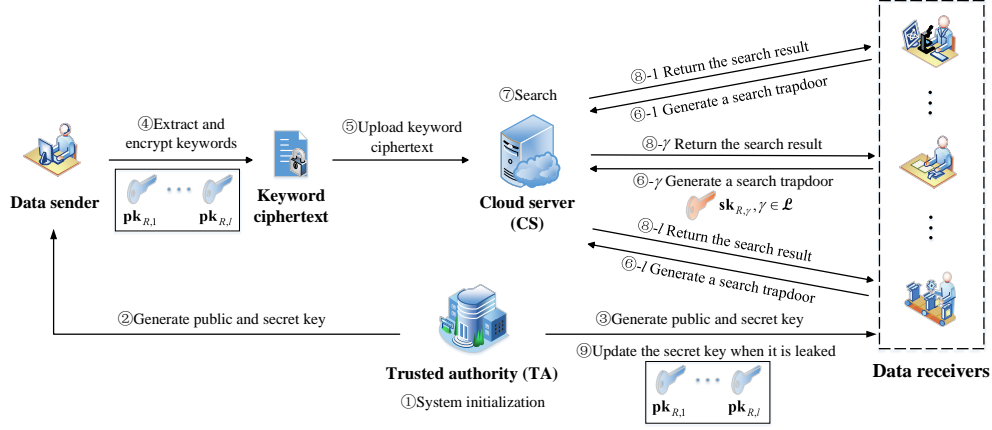


Figure 2: An outsourced data privacy-preserving framework through efficient broadcast encrypted search for cloud networks.

- $pp \leftarrow \mathbf{Setup}(1^\lambda)$ : After inputting a security parameter  $\lambda$ , this algorithm publishes a public parameter  $pp$ .
- $(\mathbf{pk}_S, \mathbf{sk}_S) \leftarrow \mathbf{KeyGen}_S(pp)$ : After inputting the public parameter  $pp$ , this algorithm publishes the public and secret keys  $(\mathbf{pk}_S, \mathbf{sk}_S)$  of the data sender.
- $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i}) \leftarrow \mathbf{KeyGen}_R(pp)$ : For  $i \in \mathcal{L}$ , after inputting the public parameter  $pp$ , this algorithm publishes the public and secret keys  $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i})$  of the data receiver  $i$ .
- $\mathbf{CT} \leftarrow \mathbf{Encrypt}(pp, \mathbf{ck}, \mathbf{pk}_S, \mathbf{sk}_S, \{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}})$ : After inputting the public parameter  $pp$ , a keyword  $\mathbf{ck} \in \mathbb{Z}_q^n$ , the public and secret keys of data sender  $(\mathbf{pk}_S, \mathbf{sk}_S)$ , the public keys of data receiver in broadcast list  $\{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}$ , the data sender invokes this algorithm to get a ciphertext  $\mathbf{CT}$ .
- $\mathbf{TD} \leftarrow \mathbf{Trapdoor}(pp, \mathbf{tk}, \{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}, \mathbf{sk}_{R,\gamma}, \mathbf{pk}_S)$ : After inputting the public parameter  $pp$ , a keyword  $\mathbf{tk} \in \mathbb{Z}_q^n$ , the public keys of data receiver in broadcast list  $\{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}$ , a secret key  $\mathbf{sk}_{R,\gamma}$  of data receiver  $\gamma$ , and the public key of data sender  $\mathbf{pk}_S$ , the data receiver  $\gamma$  invokes this algorithm to get the trapdoor  $\mathbf{TD}$ .

- $1 \text{ or } 0 \leftarrow \mathbf{Search}(\text{CT}, \text{TD})$ : The server processes this algorithm to test whether CT and TD correspond to the same keyword. If yes, it outputs 1. Otherwise, outputs 0.

Further, our FS-BroSearch is regarded as an enhanced version of BroSearch, it contains seven following algorithms  $\Pi_{\text{FS-BroSearch}} = (\mathbf{Setup}, \mathbf{KeyGen}_S, \mathbf{KeyGen}_R, \mathbf{KeyUpdate}_R, \mathbf{Encrypt}, \mathbf{Trapdoor}, \mathbf{Search})$ , which is similar as  $\Pi_{\text{BroSearch}}$ , except for the following algorithms.

- $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i,0}) \leftarrow \mathbf{KeyGen}_R(pp)$ : For  $i \in \mathcal{L}$ , after inputting the public parameter  $pp$ , this algorithm publishes the public and initial secret keys  $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i,0})$  of the data receiver  $i$ .
- $\mathbf{sk}_{R,i,t+1} \leftarrow \mathbf{KeyUpdate}_R(pp, \mathbf{pk}_{R,i}, \mathbf{sk}_{R,i,t})$ : For  $i \in \mathcal{L}$ , after inputting a public parameter  $pp$ , the public and secret keys  $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i,t})$  of data receiver with time period  $t$ , this algorithm publishes its secret key  $\mathbf{sk}_{R,i,t+1}$  with time period  $t + 1$ .
- $\text{CT}_t \leftarrow \mathbf{Encrypt}(pp, \mathbf{ck}, \mathbf{pk}_S, \mathbf{sk}_S, \{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}, t)$ : This definition is the same as the **Encrypt** algorithm in BroSearch, except for the introduction of time period  $t$ .
- $\text{TD}_t \leftarrow \mathbf{Trapdoor}(pp, \mathbf{tk}, \{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}, \mathbf{sk}_{R,\gamma,t}, \mathbf{pk}_S, t)$ : This definition is the same as the **Trapdoor** algorithm in BroSearch, except for the introduction of time period  $t$ .

#### 4.3. Security Models

We define two security models of our BroSearch: ciphertext indistinguishability against chosen keyword attacks (IND-CKA) and ciphertext indistinguishability against insider keyword guessing attacks (IND-IKGA).

##### 4.3.1. IND-CKA security

For the first part, we define the IND-CKA model  $\mathbf{Exp}_{\text{BroSearch}, \mathcal{A}}^{\text{IND-CKA}}(\lambda)$  as follows:

1. **Setup**: Given a challenge public key  $\mathbf{pk}_S^* = \mathbf{A}_S$  of data sender and several challenge secret key in broadcast list  $\{\mathbf{pk}_{R,i}^* = \mathbf{A}_{R,i}\}_{i \in \mathcal{L}^*}$ , the challenger  $\mathcal{C}$  calls the **Setup**( $1^\lambda$ ) algorithm to calculate a public parameter  $pp$ , and sends it to the adversary  $\mathcal{A}$ .
2. **Phase 1**:  $\mathcal{A}$  can adaptively perform four oracles in PPT.

- (a)  $\mathcal{O}_{H_1}$ :  $\mathcal{A}$  inputs a keyword  $\mathbf{ck}_j$  to issue  $H_1$  queries at most  $q_{H_1}$  for  $j \in [q_{H_1}]$ ,  $\mathcal{C}$  calculates  $H_1(\mathbf{ck}_j)$  and sends it to  $\mathcal{A}$ .
  - (b)  $\mathcal{O}_{H_2}$ :  $\mathcal{A}$  inputs a keyword  $\mathbf{ck}_j$  to issue  $H_2$  queries at most  $q_{H_2}$  for  $j \in [q_{H_2}]$ ,  $\mathcal{C}$  calculates  $H_2(\mathbf{ck}_j)$  and sends it to  $\mathcal{A}$ .
  - (c)  $\mathcal{O}_{CT}$ : Given a keyword  $\mathbf{ck}$  and the receivers' public keys in broadcast list  $\{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}$  from  $\mathcal{A}$ ,  $\mathcal{C}$  calls  $\mathbf{Encrypt}(pp, \mathbf{ck}, \mathbf{pk}_S^*, \mathbf{sk}_S^*, \{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}})$  algorithm to generate a ciphertext CT, and sends it to  $\mathcal{A}$ .
  - (d)  $\mathcal{O}_{TD}$ : After obtained a keyword  $\mathbf{tk}$ , the sender's public key  $\mathbf{pk}_S$  and a receiver  $\gamma \in \mathcal{L}$  from  $\mathcal{A}$ ,  $\mathcal{C}$  calls  $\mathbf{Trapdoor}(pp, \mathbf{tk}, \{\mathbf{pk}_{R,i}^*\}_{i \in \mathcal{L}^*}, \mathbf{sk}_{R,\gamma}^*, \mathbf{pk}_S)$  algorithm to generate a trapdoor TD for  $\mathcal{A}$ .
3. **Challenge**:  $\mathcal{A}$  chooses two challenge keywords  $\mathbf{ck}_0^*, \mathbf{ck}_1^*$  which have not been queried in **Phase 1**, and sends them to  $\mathcal{C}$ . After that,  $\mathcal{C}$  selects a random bit  $\xi \in \{0, 1\}$  and calls  $\mathbf{Encrypt}(pp, \mathbf{ck}_\xi^*, \mathbf{pk}_S^*, \mathbf{sk}_S^*, \{\mathbf{pk}_{R,i}^*\}_{i \in \mathcal{L}^*})$  algorithm to obtain a challenge ciphertext  $\text{CT}_\xi^*$ . Finally,  $\mathcal{C}$  returns  $\text{CT}_\xi^*$  to  $\mathcal{A}$ .
  4. **Phase 2**:  $\mathcal{A}$  executes these queries as above, neither  $\mathbf{ck}_0^*$  nor  $\mathbf{ck}_1^*$  can be queried.
  5. **Guess**:  $\mathcal{A}$  outputs a bit  $\xi' \in \{0, 1\}$ . If  $\xi' = \xi$ , we say that  $\mathcal{A}$  wins this game.

We define the advantage of  $\mathcal{A}$  to win the above game  $\mathbf{Exp}_{\text{BroSearch}, \mathcal{A}}^{\text{IND-CKA}}(\lambda)$  as:  $\text{Adv}_{\text{BroSearch}, \mathcal{A}}^{\text{IND-CKA}}(\lambda) = |\Pr[\xi' = \xi] - \frac{1}{2}|$ .

**Definition 5.** Our *BroSearch* primitive satisfies *IND-CKA* security, if any PPT malicious adversary wins the above game  $\mathbf{Exp}_{\text{BroSearch}, \mathcal{A}}^{\text{IND-CKA}}(\lambda)$  with a negligible advantage.

#### 4.3.2. IND-IKGA security

For the second part, we define the IND-IKGA security model  $\mathbf{Exp}_{\text{BroSearch}, \mathcal{A}}^{\text{IND-IKGA}}(\lambda)$  as follows:

1. **Setup**: This procedure is the same as the corresponding part in  $\mathbf{Exp}_{\text{BroSearch}, \mathcal{A}}^{\text{IND-CKA}}(\lambda)$ .
2. **Phase 1**: This procedure is the same as the corresponding part in  $\mathbf{Exp}_{\text{BroSearch}, \mathcal{A}}^{\text{IND-CKA}}(\lambda)$ .
3. **Challenge**:  $\mathcal{A}$  chooses two challenge keywords  $\mathbf{tk}_0^*, \mathbf{tk}_1^*$  which have not been queried in **Phase 1**, and sends them to  $\mathcal{C}$ . After that,  $\mathcal{C}$  selects a random bit  $\xi \in \{0, 1\}$  and calls  $\mathbf{Trapdoor}(pp, \mathbf{tk}_\xi^*, \{\mathbf{pk}_{R,i}^*\}_{i \in \mathcal{L}}, \mathbf{sk}_{R,\gamma}^*, \mathbf{pk}_S^*)$

algorithm to obtain a challenge trapdoor  $\text{TD}_\xi^*$ . Finally,  $\mathcal{C}$  returns  $\text{TD}_\xi^*$  to  $\mathcal{A}$ .

4. **Phase 2:**  $\mathcal{A}$  executes these queries as above, neither  $\text{tk}_0^*$  nor  $\text{tk}_1^*$  can be queried.
5. **Guess:**  $\mathcal{A}$  outputs a bit  $\xi' \in \{0, 1\}$ . If  $\xi' = \xi$ , we say that  $\mathcal{A}$  wins this game.

We define the advantage of  $\mathcal{A}$  to win the above game  $\text{Exp}_{\text{BroSearch}, \mathcal{A}}^{\text{IND-IKGA}}(\lambda)$  as:  $\text{Adv}_{\text{BroSearch}, \mathcal{A}}^{\text{IND-IKGA}}(\lambda) = |\Pr[\xi' = \xi] - \frac{1}{2}|$ .

**Definition 6.** *Our BroSearch primitive satisfies IND-IKGA security, if any PPT adversary wins the above game  $\text{Exp}_{\text{BroSearch}, \mathcal{A}}^{\text{IND-IKGA}}(\lambda)$  with a negligible advantage.*

Correspondingly, our FS-BroSearch also has two security models,  $\text{Exp}_{\text{FS-BroSearch}, \mathcal{A}}^{\text{IND-CKA}}(\lambda)$  and  $\text{Exp}_{\text{FS-BroSearch}, \mathcal{A}}^{\text{IND-IKGA}}(\lambda)$ , which are highly symmetric with respect to  $\text{Exp}_{\text{BroSearch}, \mathcal{A}}^{\text{IND-CKA}}(\lambda)$  and  $\text{Exp}_{\text{BroSearch}, \mathcal{A}}^{\text{IND-IKGA}}(\lambda)$ , except for the addition of the  $\mathcal{O}_{\text{KU}}$  oracle and the introduction of time periods.  $\mathcal{O}_{\text{KU}}$  is defined as follows:

- $\mathcal{O}_{\text{KU}}$ : After obtaining a time period  $t$  for  $i \in \mathcal{L}$  from  $\mathcal{A}$ ,  $\mathcal{C}$  calls  $\text{KeyUpdate}_R(pp, \text{pk}_{R,i}, \text{sk}_{R,i,t-1})$  algorithm to calculate the secret key  $\text{sk}_{R,i,t}$  with time period  $t$ , and sends it to  $\mathcal{A}$ .

Based on this, we define the IND-CKA and IND-IKGA security of our FS-BroSearch as follows:

**Definition 7.** *Our FS-BroSearch primitive satisfies IND-CKA security, if any PPT malicious adversary wins the above game  $\text{Exp}_{\text{FS-BroSearch}, \mathcal{A}}^{\text{IND-CKA}}(\lambda)$  with a negligible advantage.*

**Definition 8.** *Our FS-BroSearch primitive satisfies IND-IKGA security, if any PPT adversary wins the above game  $\text{Exp}_{\text{FS-BroSearch}, \mathcal{A}}^{\text{IND-IKGA}}(\lambda)$  with a negligible advantage.*

## 5. The Design of BroSearch

### 5.1. Concrete Construction

The construction of our BroSearch includes five procedures: System Initialization, Key Generation, Ciphertext Calculation, Trapdoor Generation, and Search.

### 5.1.1. System Initialization

TA initializes the entire system by calling the **Setup**( $1^\lambda$ ) algorithm through inputting a security parameter  $1^\lambda$ . Firstly, it sets several system parameters  $n, m, q, \sigma, l$ , a gadget matrix  $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ , and a broadcast list  $\mathcal{L} = \{1, \dots, l\}$ . Then, TA chooses two matrices  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$  and  $\mathbf{U} \in \mathbb{Z}_q^{n \times n}$  uniformly. To ensure the security of our realization in the ROM, two hash functions are defined as  $H_1 : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q^{n \times m}$ ,  $H_2 : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q^{n \times n}$ . Finally, TA obtains the public parameter and sends it to other entities. The public parameter is defined as:

$$pp := (n, m, q, \sigma, l, \mathbf{G}, \mathcal{L}, \mathbf{A}, \mathbf{U}, H_1, H_2).$$

### 5.1.2. Key Generation

This procedure generates public and secret keys for the data sender and receiver through **KeyGen<sub>S</sub>**( $pp$ ) and **KeyGen<sub>R</sub>**( $pp$ ) algorithms, respectively.

For the data sender, TA invokes  $(\mathbf{A}_S, \mathbf{T}_{\mathbf{A}_S}) \leftarrow \text{TrapGen}(n, m, q)$  to generate an uniformly matrix  $\mathbf{A}_S \in \mathbb{Z}_q^{n \times m}$  and a basis  $\mathbf{T}_{\mathbf{A}_S} \in \mathbb{Z}^{m \times m}$  of  $\Lambda_q^\perp(\mathbf{A}_S)$ . After that, TA obtains the public and secret keys of the data sender:

$$\mathbf{pk}_S := \mathbf{A}_S, \mathbf{sk}_S := \mathbf{T}_{\mathbf{A}_S}.$$

Similarly, for the data receiver  $i \in \mathcal{L}$ , TA calls  $(\mathbf{A}_{R,i}, \mathbf{T}_{\mathbf{A}_{R,i}}) \leftarrow \text{TrapGen}(n, m, q)$  to generate an uniformly matrix  $\mathbf{A}_{R,i} \in \mathbb{Z}_q^{n \times m}$  and a basis  $\mathbf{T}_{\mathbf{A}_{R,i}} \in \mathbb{Z}^{m \times m}$  of  $\Lambda_q^\perp(\mathbf{A}_{R,i})$ , and defines the public and secret keys of the data receiver  $i$ :

$$\mathbf{pk}_{R,i} := \mathbf{A}_{R,i}, \mathbf{sk}_{R,i} := \mathbf{T}_{\mathbf{A}_{R,i}}.$$

At last, TA returns  $(\mathbf{pk}_S, \mathbf{sk}_S)$  and  $\{(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i})\}_{i \in \mathcal{L}}$  to a data sender and several data receivers in broadcast list  $\mathcal{L}$  over a secure channel.

### 5.1.3. Ciphertext Calculation

To calculate the keyword ciphertext, the data sender calls the **Encrypt**( $pp, \mathbf{ck}, \mathbf{pk}_S, \mathbf{sk}_S, \{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}$ ) algorithm. In this procedure, the data sender inputs its secret key to authenticate the keyword  $\mathbf{ck}$ , which can resist IKGAs. On the other hand, the public keys of all the data receivers in the broadcast list  $\mathcal{L}$  are embedded to enable the ciphertext broadcast.

Specifically, the data sender chooses a vector  $\mathbf{s} \in \mathbb{Z}_q^n$  and several error vectors  $\{\mathbf{e}_{R,i}\}_{i \in \mathcal{L}} \in \chi^m$ ,  $\mathbf{e}_K \in \chi^m$ ,  $\mathbf{e}_U \in \chi^n$ . For  $i \in \mathcal{L}$ , it calculates a vector  $\mathbf{c}_{R,i} = \mathbf{A}_{R,i}^\top \mathbf{s} + \mathbf{e}_{R,i} \in \mathbb{Z}_q^m$  to embed public keys of receivers in  $\mathcal{L}$ . Subsequently, the data sender calculate two vectors  $\mathbf{c}_K = H_1(\mathbf{ck})^\top \mathbf{s} + \mathbf{e}_K \in \mathbb{Z}_q^m$  and  $\mathbf{c}_U =$

$\mathbf{U}^\top \mathbf{s} + \mathbf{e}_U \in \mathbb{Z}_q^n$ . To authenticate the keyword  $\mathbf{ck}$ , it invokes the following algorithm to sample a vector  $\boldsymbol{\varepsilon}_C \in \mathbb{Z}_q^{2m}$ , where  $\boldsymbol{\varepsilon}_C$  is statistically distributed in  $\mathcal{D}_{\Lambda_q^{\mathbf{cU}}(\mathbf{A}_S | (\mathbf{A} + H_2(\mathbf{ck})\mathbf{G}))}^{2m}$ .

$$\boldsymbol{\varepsilon}_C \leftarrow \text{SampleLeft}(\mathbf{A}_S, \mathbf{A} + H_2(\mathbf{ck})\mathbf{G}, \mathbf{T}_{\mathbf{A}_S}, \mathbf{c}_U, \sigma),$$

such that  $(\mathbf{A}_S | (\mathbf{A} + H_2(\mathbf{ck})\mathbf{G}))\boldsymbol{\varepsilon}_C = \mathbf{c}_U \bmod q$ .

To sum up, the data sender outputs the following ciphertext with the keyword  $\mathbf{ck}$ , and uploads it to CS.

$$\text{CT} := (\{\mathbf{c}_{R,i}\}_{i \in \mathcal{L}}, \mathbf{c}_K, \boldsymbol{\varepsilon}_C).$$

#### 5.1.4. Trapdoor Generation

A data receiver  $\gamma \in \mathcal{L}$  performs the **Trapdoor**( $pp, \mathbf{tk}, \{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}, \mathbf{sk}_{R,\gamma}, \mathbf{pk}_S$ ) algorithm to generate a trapdoor with an interested keyword  $\mathbf{tk}$ . This procedure is divided into two phases: the keyword embedding and the receiver's secret key processing.

For the keyword  $\mathbf{tk}$ , the data receiver chooses a vector  $\mathbf{s}' \in \mathbb{Z}_q^n$ , several error vectors  $\mathbf{e}_S \in \chi^m$ ,  $\mathbf{e}'_U \in \chi^n$ , and a matrix  $\mathbf{R}'_K \in \{-1, 1\}^{m \times m}$ . Next, it calculates three vectors  $\mathbf{t}_S = \mathbf{A}_S^\top \mathbf{s}' + \mathbf{e}_S \in \mathbb{Z}_q^m$ ,  $\mathbf{t}_K = (\mathbf{A} + H_2(\mathbf{tk})\mathbf{G})^\top \mathbf{s}' + (\mathbf{R}'_K)^\top \mathbf{e}_S \in \mathbb{Z}_q^m$  and  $\mathbf{t}_U = \mathbf{U}\mathbf{s}' + \mathbf{e}'_U \in \mathbb{Z}_q^n$  to embed this keyword.

Moreover, the data receiver generates a vector  $\boldsymbol{\varepsilon}_T$  with its secret key  $\mathbf{T}_{\mathbf{A}_{R,\gamma}}$  through the following algorithm:

$$\boldsymbol{\varepsilon}_T \leftarrow \text{GenSamplePre}(\mathbf{A}_{R,1} | \cdots | \mathbf{A}_{R,l} | H_1(\mathbf{tk}), \mathbf{T}_{\mathbf{A}_{R,\gamma}}, \{\gamma\}, \mathbf{t}_u, \sigma),$$

such that  $(\mathbf{A}_{R,1} | \cdots | \mathbf{A}_{R,l} | H_1(\mathbf{tk}))\boldsymbol{\varepsilon}_T = \mathbf{t}_U \bmod q$ , where  $\boldsymbol{\varepsilon}_T$  is statistically distributed in  $\mathcal{D}_{\Lambda_q^{\mathbf{tU}}(\mathbf{A}_{R,1} | \cdots | \mathbf{A}_{R,l} | H_1(\mathbf{tk}))}^{(l+1)m}$ .

Eventually, the data receiver sends the following trapdoor to CS.

$$\text{TD} := (\mathbf{t}_S, \mathbf{t}_K, \boldsymbol{\varepsilon}_T).$$

If the data receiver is in the broadcast list, it can generate the trapdoor used for the encrypted search by inputting its secret key. Otherwise, the data receiver cannot search on it, since the basis  $\mathbf{T}_{\mathbf{A}_{R,\gamma}}$  does not match the matrix  $(\mathbf{A}_{R,1} | \cdots | \mathbf{A}_{R,l} | H_1(\mathbf{tk}))$  in **GenSamplePre** algorithm.

### 5.1.5. Search

After receiving the trapdoor TD, CS iterates over the keywords CT uploaded on it, and calls the **Search**(CT, TD) algorithm to obtain a search result.

In this procedure, CS parses  $CT = (\{\mathbf{c}_{R,i}\}_{i \in \mathcal{L}}, \mathbf{c}_K, \boldsymbol{\varepsilon}_C)$  and  $TD = (\mathbf{t}_S, \mathbf{t}_K, \boldsymbol{\varepsilon}_T)$ , and calculates a number  $d = \boldsymbol{\varepsilon}_C^\top (\mathbf{t}_S^\top \parallel \mathbf{t}_K^\top)^\top - (\mathbf{c}_{R,1}^\top \parallel \cdots \parallel \mathbf{c}_{R,l}^\top \parallel \mathbf{c}_K^\top) \boldsymbol{\varepsilon}_T \in \mathbb{Z}_q$ .

If  $|d| < \lfloor \frac{q}{4} \rfloor$ , this procedure outputs 1, meaning that CT and TD correspond to the same keyword, and returns the search result to the data receiver. Otherwise, outputs 0.

### 5.2. Correctness Analysis

We analyze the correctness of our BroSearch. Given the data sender's public and secret keys  $\mathbf{pk}_S = \mathbf{A}_S, \mathbf{sk}_S = \mathbf{T}_{\mathbf{A}_S}$ , a keyword  $\mathbf{ck} \in \mathbb{Z}_q^n$ , and its ciphertext  $CT = (\{\mathbf{c}_{R,i}\}_{i \in \mathcal{L}}, \mathbf{c}_K, \boldsymbol{\varepsilon}_C)$ . Moreover, the data receiver  $\gamma$  owns its public keys and secret keys  $\mathbf{pk}_{R,\gamma} := \mathbf{A}_{R,\gamma}, \mathbf{sk}_{R,\gamma} := \mathbf{T}_{\mathbf{A}_{R,\gamma}}$ , and the searched keyword  $\mathbf{tk} \in \mathbb{Z}_q^n$ , and corresponding search trapdoor  $TD = (\mathbf{t}_S, \mathbf{t}_K, \boldsymbol{\varepsilon}_T)$ . If  $\mathbf{ck} = \mathbf{tk}$ , we have:

$$\begin{aligned}
d &= \boldsymbol{\varepsilon}_C^\top (\mathbf{t}_S^\top \parallel \mathbf{t}_K^\top)^\top - (\mathbf{c}_{R,1}^\top \parallel \cdots \parallel \mathbf{c}_{R,l}^\top \parallel \mathbf{c}_K^\top) \boldsymbol{\varepsilon}_T \\
&= \boldsymbol{\varepsilon}_C^\top ((\mathbf{A}_S^\top \mathbf{s}' + \mathbf{e}_S)^\top \parallel ((\mathbf{A} + H_2(\mathbf{tk})\mathbf{G})^\top \mathbf{s}' + \mathbf{R}'_K \mathbf{e}_S)^\top)^\top \\
&\quad - ((\mathbf{A}_{R,1}^\top \mathbf{s} + \mathbf{e}_{R,1})^\top \parallel \cdots \parallel (\mathbf{A}_{R,l}^\top \mathbf{s} + \mathbf{e}_{R,l})^\top \parallel (H_1(\mathbf{ck})^\top \mathbf{s} + \mathbf{e}_K)^\top) \boldsymbol{\varepsilon}_T \\
&= \boldsymbol{\varepsilon}_C^\top (\mathbf{A}_S \parallel (\mathbf{A} + H_2(\mathbf{tk})\mathbf{G}))^\top \mathbf{s}' + \boldsymbol{\varepsilon}_C^\top (\mathbf{e}_S^\top \parallel \mathbf{e}_S^\top \mathbf{R}'_K) \\
&\quad - \mathbf{s}^\top (\mathbf{A}_{R,1} \parallel \cdots \parallel \mathbf{A}_{R,l} \parallel H_1(\mathbf{ck})) \boldsymbol{\varepsilon}_T - (\mathbf{e}_{R,1}^\top \parallel \cdots \parallel \mathbf{e}_{R,l}^\top \parallel \mathbf{e}_K^\top) \boldsymbol{\varepsilon}_T \\
&= \mathbf{c}_U^\top \mathbf{s}' - \mathbf{s}^\top \mathbf{t}_U + \boldsymbol{\varepsilon}_C^\top (\mathbf{e}_S^\top \parallel \mathbf{e}_S^\top \mathbf{R}'_K) - (\mathbf{e}_{R,1}^\top \parallel \cdots \parallel \mathbf{e}_{R,l}^\top \parallel \mathbf{e}_K^\top) \boldsymbol{\varepsilon}_T \\
&= \mathbf{s}^\top \mathbf{U} \mathbf{s}' + \mathbf{e}_U^\top \mathbf{s}' - \mathbf{s}^\top \mathbf{U} \mathbf{s}' - \mathbf{s}^\top \mathbf{e}'_U + \boldsymbol{\varepsilon}_C^\top (\mathbf{e}_S^\top \parallel \mathbf{e}_S^\top \mathbf{R}'_K) - (\mathbf{e}_{R,1}^\top \parallel \cdots \parallel \mathbf{e}_{R,l}^\top \parallel \mathbf{e}_K^\top) \boldsymbol{\varepsilon}_T \\
&= \mathbf{e}_U^\top \mathbf{s}' - \mathbf{s}^\top \mathbf{e}'_U + \boldsymbol{\varepsilon}_C^\top (\mathbf{e}_S^\top \parallel \mathbf{e}_S^\top \mathbf{R}'_K) - (\mathbf{e}_{R,1}^\top \parallel \cdots \parallel \mathbf{e}_{R,l}^\top \parallel \mathbf{e}_K^\top) \boldsymbol{\varepsilon}_T \\
&= \text{error},
\end{aligned}$$

$$\text{where } |\text{error}| \leq |\mathbf{e}_U^\top \mathbf{s}'| + |\mathbf{s}^\top \mathbf{e}'_U| + |\boldsymbol{\varepsilon}_C^\top (\mathbf{e}_S^\top \parallel \mathbf{e}_S^\top \mathbf{R}'_K)| + |(\mathbf{e}_{R,1}^\top \parallel \cdots \parallel \mathbf{e}_{R,l}^\top \parallel \mathbf{e}_K^\top) \boldsymbol{\varepsilon}_T| \leq 2n\chi_m^2 + 40\sigma m^{\frac{3}{2}}\chi_m + (l+1)\sigma m\chi_m < \frac{q}{5}.$$

### 5.3. Parameters Setting

In this section, we provide the parameter settings as:

- $m \geq \lceil 2n \log q \rceil$  for the TrapGen lemma.



- $\sigma \geq O(\sqrt{n \log q}) \cdot \omega(\sqrt{\log(l+1)m})$  for **SampleLeft** and **GenSamplePre** lemmas.
- $\chi_m \geq \sqrt{n} \cdot \omega(\log n)$  for **LWE hardness**.
- $2n\chi_m^2 + 40\sigma m^{\frac{3}{2}}\chi_m + (l+1)\sigma m\chi_m < \frac{q}{5}$ .

#### 5.4. Security Analysis

We demonstrate that our **BroSearch** is secure in the aforementioned security model, i.e., **IND-CKA** and **IND-IKGA**.

**Theorem 1.** *Assuming that the  $\text{LWE}_{n,m,q,\chi}$  hardness holds, our proposed **BroSearch** satisfies **IND-CKA** security in the random oracle model. For any PPT adversary  $\mathcal{A}$ , if  $\mathcal{A}$  can compromise our scheme with a non-negligible advantage  $\epsilon_1$ , then we can construct a PPT challenger  $\mathcal{C}$  to solve the  $\text{LWE}_{n,m,q,\chi}$  hardness with a non-negligible probability.*

**Proof 1.** *If a PPT adversary  $\mathcal{A}$  can compromise the **IND-CKA** security with a non-negligible advantage, we can construct a challenger  $\mathcal{C}$  who can solve the  $\text{LWE}_{n,m,q,\chi}$  hardness. The following procedures show the interaction between  $\mathcal{A}$  and  $\mathcal{C}$ .*

**LWE Instances:**  $\mathcal{C}$  obtains several **LWE** instances  $\{\mathbf{A}_{R,i}, \mathbf{a}_{R,i}\}_{i \in \mathcal{L}}, (\mathbf{A}_K, \mathbf{a}_K)$ , which are sampled from either  $\mathcal{O}_S$  or  $\mathcal{O}_\S$ . If we sample them from  $\mathcal{O}_S$ , they satisfy  $\{\mathbf{a}_{R,i} = \mathbf{A}_{R,i}^\top \mathbf{s} + \mathbf{e}_{R,i}\}_{i \in \mathcal{L}} \in \mathbb{Z}_q^m$  and  $\mathbf{a}_K = \mathbf{A}_K^\top \mathbf{s} + \mathbf{e}_K \in \mathbb{Z}_q^m$  for a secret vector  $\mathbf{s} \in \mathbb{Z}_q^n$  and several error vectors  $\{\mathbf{e}_{R,i}\}_{i \in \mathcal{L}} \in \mathbb{Z}_q^m$  and  $\mathbf{e}_K \in \mathbb{Z}_q^m$ . Otherwise, they are random over  $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$ .

**Init:**  $\mathcal{C}$  selects a challenge public key  $\mathbf{pk}_S^* = \mathbf{A}_S$  of data sender and several challenge secret key in broadcast list  $\{\mathbf{pk}_{R,i}^* = \mathbf{A}_{R,i}\}_{i \in \mathcal{L}^*}$ .

**Setup:**  $\mathcal{C}$  obtains the public parameter  $pp$  through calling **Setup**( $1^\lambda$ ) algorithm, except that  $\mathbf{A} = \mathbf{A}_S \mathbf{R}_K^* - h^* \mathbf{G} \in \mathbb{Z}_q^{n \times m}$  for a random value  $h^* \in \mathbb{Z}_q$  and a matrix  $\mathbf{R}_K^* \in \{-1, 1\}^{m \times m}$ , and then sends it to  $\mathcal{A}$ .

**Phase 1:**  $\mathcal{A}$  executes these following queries adaptively:

- $\mathcal{O}_{H_1}$ :  $\mathcal{A}$  inputs a keyword  $\mathbf{ck}_j$  to issue  $H_1$  queries at most  $q_{H_1}$  for  $j \in [q_{H_1}]$ . Initially,  $\mathcal{C}$  creates a empty list  $L_{H_1}$ , and selects  $\pi_1^* \in [q_{H_1}]$  as a challenge query. For the  $j$ -th query, if  $j = \pi_1^*$ ,  $\mathcal{C}$  sets  $H_1(\mathbf{ck}_j) = \mathbf{A}_K$ , selects  $\mathbf{T}_{H_1(\mathbf{ck}_j)} \in \mathbb{Z}^{m \times m}$  randomly. Then,  $\mathcal{C}$  adds  $\{\mathbf{ck}_j, H_1(\mathbf{ck}_j), \mathbf{T}_{H_1(\mathbf{ck}_j)}\}$  to  $L_{H_1}$ , and then returns  $H_1(\mathbf{ck}_j)$  to  $\mathcal{A}$ . Otherwise, if  $\mathbf{ck}_j$  has been queried,  $\mathcal{C}$  returns  $H_1(\mathbf{ck}_j)$  in  $L_{H_1}$  to  $\mathcal{A}$ . Otherwise,  $\mathcal{C}$  invokes **TrapGen**( $n, m, q$ ) to generate a matrix  $\mathbf{A}_H \in \mathbb{Z}_q^{n \times m}$  and

a basis  $\mathbf{T}_{\mathbf{A}_H} \in \mathbb{Z}^{m \times m}$  of  $\Lambda_q^\perp(\mathbf{A}_H)$ . Subsequently,  $\mathcal{C}$  sets  $H_1(\mathbf{ck}_j) = \mathbf{A}_H$  and  $\mathbf{T}_{H_1(\mathbf{ck}_j)} = \mathbf{T}_{\mathbf{A}_H}$ , sends  $H_1(\mathbf{ck}_j)$  to  $\mathcal{A}$ , and then adds  $\{\mathbf{ck}_j, H_1(\mathbf{ck}_j), \mathbf{T}_{H_1(\mathbf{ck}_j)}\}$  to  $L_{H_1}$ .

- $\mathcal{O}_{H_2}$ : In this phase,  $\mathcal{A}$  issues  $H_2$  queries at most  $q_{H_2}$  after inputting a keyword  $\mathbf{ck}_j$  for  $j \in [q_{H_2}]$ .  $\mathcal{C}$  first creates a empty list  $L_{H_2}$ , and selects  $\pi_2^* \in [q_{H_2}]$  as a challenge query. For the  $j$ -th query, if  $j = \pi_2^*$ ,  $\mathcal{C}$  sets  $H_2(\mathbf{ck}_j) = h^*$ , adds  $\{\mathbf{ck}_j, H_2(\mathbf{ck}_j)\}$  to  $L_{H_2}$ , and returns  $h^*$  to  $\mathcal{A}$ . Otherwise, if  $\mathbf{ck}_j$  has been queried,  $\mathcal{C}$  returns  $H_2(\mathbf{ck}_j)$  in  $L_{H_2}$  to  $\mathcal{A}$ . Otherwise,  $\mathcal{C}$  chooses a random value  $h \in \mathbb{Z}_q$  as  $H_2(\mathbf{ck}_j)$ , sends it to  $\mathcal{A}$ , and adds  $\{\mathbf{ck}_j, H_2(\mathbf{ck}_j)\}$  to  $L_{H_2}$ .
- $\mathcal{O}_{CT}$ : After obtaining a keyword  $\mathbf{ck}$  and the receivers' public keys in broadcast list  $\{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}$  from  $\mathcal{A}$ ,  $\mathcal{C}$  calls **Encrypt**( $pp, \mathbf{ck}, \mathbf{pk}_S^*, \mathbf{sk}_S^*, \{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}$ ) algorithm to generate a ciphertext CT, and sends it to  $\mathcal{A}$ .
- $\mathcal{O}_{TD}$ : After obtaining a keyword  $\mathbf{tk}$ , the sender's public key  $\mathbf{pk}_S$  and a receiver  $\gamma \in \mathcal{L}$  from  $\mathcal{A}$ ,  $\mathcal{C}$  executes the  $\mathcal{O}_{H_1}$  and  $\mathcal{O}_{H_2}$  oracle to query the hash value of  $\mathbf{tk}$ . If  $H_1(\mathbf{tk}) = \mathbf{A}_K$  or  $H_2(\mathbf{tk}) = h^*$ , this procedure is aborted by  $\mathcal{C}$ . Otherwise,  $\mathcal{C}$  obtains  $\{\mathbf{tk}, H_1(\mathbf{tk}), \mathbf{T}_{H_1(\mathbf{tk})}\}$  and  $\{\mathbf{tk}, H_2(\mathbf{tk})\}$  in  $L_{H_1}$  and  $L_{H_2}$ . Then,  $\mathcal{C}$  calculates  $\mathbf{t}_S$ ,  $\mathbf{t}_K$  and  $\mathbf{t}_U$  as in **Trapdoor** algorithm, and invokes  $\boldsymbol{\varepsilon}_T \leftarrow \text{GenSamplePre}(\mathbf{A}_{R,1} \mid \cdots \mid \mathbf{A}_{R,l} \mid H_1(\mathbf{tk}), \mathbf{T}_{H_1(\mathbf{tk})}, \{l+1\}, \mathbf{t}_U, \sigma)$  such that  $(\mathbf{A}_{R,1} \mid \cdots \mid \mathbf{A}_{R,l} \mid H_1(\mathbf{tk}))\boldsymbol{\varepsilon}_T = \mathbf{t}_U \bmod q$ . Finally,  $\mathcal{C}$  returns  $\text{TD} = (\mathbf{t}_S, \mathbf{t}_K, \boldsymbol{\varepsilon}_T)$  to  $\mathcal{A}$ .

**Challenge:**  $\mathcal{A}$  transmits  $\mathbf{ck}_0^*, \mathbf{ck}_1^* \in \mathbb{Z}_q^n$  to  $\mathcal{C}$ , which have not been queried in **Phase 1**. Then,  $\mathcal{C}$  chooses  $\xi \in \{0, 1\}$ , and calculates a challenge ciphertext  $\text{CT}_\xi^* = (\{\mathbf{c}_{R,i}^*\}_{i \in \mathcal{L}^*}, \mathbf{c}_K^*, \boldsymbol{\varepsilon}_C^*)$ , where  $\mathbf{c}_{R,i}^* = \mathbf{a}_{R,i}$  for  $i \in \mathcal{L}^*$ ,  $\mathbf{c}_K^* = \mathbf{a}_K$  and  $\boldsymbol{\varepsilon}_C^* \leftarrow \mathbb{Z}_q^{2m}$ . After that,  $\mathcal{C}$  returns  $\text{CT}_\xi^*$  to  $\mathcal{A}$ .

**Phase 2:**  $\mathcal{A}$  executes these queries as above, and promises neither  $\mathbf{ck}_0^*$  nor  $\mathbf{ck}_1^*$  can be queried.

**Guess:**  $\mathcal{A}$  outputs a random bit  $\xi' \in \{0, 1\}$  after receiving  $\text{CT}_\xi^*$ . If  $\xi' = \xi$ ,  $\mathcal{A}$  wins this game, and  $\mathcal{C}$  outputs 1. Otherwise,  $\mathcal{C}$  outputs 0.

**Analysis:** We demonstrate two cases according to the sampling way of LWE instances. If these instances are sampled from the  $\mathcal{O}_S$ , we have:

$$\begin{aligned} \mathbf{c}_{R,i}^* &= \mathbf{a}_{R,i} = \mathbf{A}_{R,i}^\top \mathbf{s} + \mathbf{e}_{R,i}, \text{ where } i \in \mathcal{L}^*, \\ \mathbf{c}_K^* &= \mathbf{a}_K = \mathbf{A}_K^\top \mathbf{s} + \mathbf{e}_K = H_1(\mathbf{ck}_\xi^*)^\top \mathbf{s} + \mathbf{e}_K, \\ \boldsymbol{\varepsilon}_C^* &\leftarrow \mathbb{Z}_q^{2m}. \end{aligned}$$

Thus,  $\text{CT}_\xi^* = (\{\mathbf{c}_{R,i}^*\}_{i \in \mathcal{L}^*}, \mathbf{c}_K^*, \boldsymbol{\varepsilon}_C^*)$  is a valid ciphertext. When  $\mathcal{A}$  compromise our scheme with advantage  $\epsilon_1$ ,  $\Pr[\xi' = \xi | \mathcal{O}_{\text{LWE}} = \mathcal{O}_S] = \frac{1}{2} + \epsilon_1$ . Otherwise,  $\text{CT}_\xi^*$  is uniform over  $(\mathbb{Z}_q^m)^l \times \mathbb{Z}_q^m \times \mathbb{Z}_q^{2m}$ . In this case,  $\Pr[\xi' = \xi | \mathcal{O}_{\text{LWE}} = \mathcal{O}_\S] = \frac{1}{2}$ . Due to the probability of executing this procedure with probability  $\frac{q_{H_1} + q_{H_2} - 1}{q_{H_1} q_{H_2}}$ , the advantage of  $\mathcal{C}$  to solve the  $\text{LWE}_{n,m,q,\chi}$  hardness is  $\frac{(q_{H_1} + q_{H_2} - 1)\epsilon_1}{2q_{H_1} q_{H_2}}$ , which is also non-negligible.

**Theorem 2.** Assuming that the  $\text{LWE}_{n,m,q,\chi}$  hardness holds, our proposed BroSearch satisfies IND-IKGA security in the random oracle model. For any PPT adversary  $\mathcal{A}$ , if  $\mathcal{A}$  can compromise our scheme with a non-negligible advantage  $\epsilon_2$ , then we can construct a PPT challenger  $\mathcal{C}$  to solve the  $\text{LWE}_{n,m,q,\chi}$  hardness with a non-negligible probability.

**Proof 2.** If a PPT adversary  $\mathcal{A}$  can compromise the IND-IKGA security with a non-negligible advantage, we can construct a challenger  $\mathcal{C}$  who can solve the  $\text{LWE}_{n,m,q,\chi}$  hardness. The following procedures show the interaction between  $\mathcal{A}$  and  $\mathcal{C}$ .

**LWE Instances:** This procedure is the same as Theorem 1, except that  $(\mathbf{A}_S, \mathbf{a}_S)$  and  $(\mathbf{A}_K, \mathbf{a}_K)$  are sampled as LWE instances in this phase.

**Init:**  $\mathcal{C}$  selects a challenge public key  $\mathbf{pk}_S^* = \mathbf{A}_S$  of data sender and a challenge secret key  $\mathbf{pk}_{R,\gamma}^* = \mathbf{A}_{R,\gamma}$  of data receiver where  $\gamma \in \mathcal{L}^*$ .

**Setup:** This procedure is the same as Theorem 1.

**Phase 1:**  $\mathcal{A}$  executes these following queries adaptively:

- $\mathcal{O}_{H_1}$ : This procedure is the same as Theorem 1.
- $\mathcal{O}_{H_2}$ : This procedure is the same as Theorem 1.
- $\mathcal{O}_{\text{CT}}$ : After obtaining a keyword  $\mathbf{ck}$  and the receivers' public keys  $\{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}$  from  $\mathcal{A}$ ,  $\mathcal{C}$  executes the  $\mathcal{O}_{H_1}$  and  $\mathcal{O}_{H_2}$  oracle to query the hash value of  $\mathbf{tk}$ . If  $H_1(\mathbf{tk}) = \mathbf{A}_K$  or  $H_2(\mathbf{tk}) = h^*$ , this procedure is aborted by  $\mathcal{C}$ . Otherwise,  $\mathcal{C}$  obtains  $\{\mathbf{tk}, H_1(\mathbf{tk}), \mathbf{T}_{H_1(\mathbf{tk})}\}$  and  $\{\mathbf{tk}, H_2(\mathbf{tk})\}$  in  $L_{H_1}$  and  $L_{H_2}$ . Then,  $\mathcal{C}$  calculates  $\{\mathbf{c}_{R,i}\}_{i \in \mathcal{L}}$ ,  $\mathbf{c}_K$  and  $\mathbf{c}_U$  as in **Encrypt** algorithm, and invokes  $\boldsymbol{\varepsilon}_C \leftarrow \text{SampleRight}(\mathbf{A}_S, (H_2(\mathbf{ck}) - h^*)\mathbf{G}, \mathbf{R}_K^*, \mathbf{T}_G, \mathbf{c}_U, \sigma)$  such that  $(\mathbf{A}_S \mid (\mathbf{A} + H_2(\mathbf{ck})\mathbf{G}))\boldsymbol{\varepsilon}_C = \mathbf{c}_U \bmod q$ . Finally,  $\mathcal{C}$  returns  $\text{CT} = (\{\mathbf{c}_{R,i}\}_{i \in \mathcal{L}}, \mathbf{c}_K, \boldsymbol{\varepsilon}_C)$  to  $\mathcal{A}$ .
- $\mathcal{O}_{\text{TD}}$ : After obtaining a keyword  $\mathbf{tk}$ , the sender's public key  $\mathbf{pk}_S$  and a receiver  $\gamma \in \mathcal{L}$  from  $\mathcal{A}$ ,  $\mathcal{C}$  calls **Trapdoor**( $pp, \mathbf{tk}, \{\mathbf{pk}_{R,i}^*\}_{i \in \mathcal{L}}, \mathbf{sk}_{R,\gamma}^*, \mathbf{pk}_S$ ) algorithm to generate a trapdoor TD, and sends it to  $\mathcal{A}$ .

**Challenge:**  $\mathcal{A}$  transmits  $\mathbf{tk}_0^*, \mathbf{tk}_1^* \in \mathbb{Z}_q^n$  to  $\mathcal{C}$ , which have not been queried in **Phase 1**. Then,  $\mathcal{C}$  chooses  $\xi \in \{0, 1\}$ , and calculates a challenge ciphertext  $\text{TD}_\xi^* = (\mathbf{t}_S^*, \mathbf{t}_K^*, \boldsymbol{\varepsilon}_T^*)$ , where  $\mathbf{t}_S^* = \mathbf{a}_S$ ,  $\mathbf{t}_K^* = \begin{pmatrix} \mathbf{a}_S \\ (\mathbf{R}_K^*)^\top \mathbf{a}_S \end{pmatrix}$  and  $\boldsymbol{\varepsilon}_T^* \leftarrow \mathbb{Z}_q^{(l+1)m}$ . After that,  $\mathcal{C}$  returns  $\text{TD}_\xi^*$  to  $\mathcal{A}$ .

**Phase 2:**  $\mathcal{A}$  executes these queries as above, and promises neither  $\mathbf{tk}_0^*$  nor  $\mathbf{tk}_1^*$  can be queried.

**Guess:**  $\mathcal{A}$  outputs a random bit  $\xi' \in \{0, 1\}$  after receiving  $\text{TD}_\xi^*$ . If  $\xi' = \xi$ ,  $\mathcal{A}$  wins this game, and  $\mathcal{C}$  outputs 1. Otherwise,  $\mathcal{C}$  outputs 0.

**Analysis:** We demonstrate two cases according to the sampling way of LWE instances. If these instances are sampled from the  $\mathcal{O}_S$ , we have:

$$\begin{aligned} \mathbf{t}_S^* &= \mathbf{a}_S = \mathbf{A}_S^\top \mathbf{s} + \mathbf{e}_S, \\ \mathbf{t}_K^* &= \begin{pmatrix} \mathbf{a}_S \\ (\mathbf{R}_K^*)^\top \mathbf{a}_S \end{pmatrix} = \begin{pmatrix} \mathbf{A}_S^\top \mathbf{s} + \mathbf{e}_S \\ (\mathbf{A}_S \mathbf{R}_K^*)^\top \mathbf{s} + (\mathbf{R}_K^*)^\top \mathbf{e}_S \end{pmatrix} \\ &= (\mathbf{A}_S \mid \mathbf{A} + H_2(\mathbf{tk}_\xi^*) \mathbf{G})^\top \mathbf{s} + (\mathbf{e}_S^\top \mid \mathbf{e}_S^\top \mathbf{R}_K^*)^\top, \\ \boldsymbol{\varepsilon}_T^* &\leftarrow \mathbb{Z}_q^{(l+1)m}. \end{aligned}$$

Thus,  $\text{TD}_\xi^* = (\mathbf{t}_S^*, \mathbf{t}_K^*, \boldsymbol{\varepsilon}_T^*)$  is a valid trapdoor. When  $\mathcal{A}$  compromise our scheme with advantage  $\epsilon_2$ ,  $\Pr[\xi' = \xi \mid \mathcal{O}_{\text{LWE}} = \mathcal{O}_S] = \frac{1}{2} + \epsilon_2$ . Otherwise,  $\text{TD}_\xi^*$  is uniform over  $\mathbb{Z}_q^m \times \mathbb{Z}_q^m \times \mathbb{Z}_q^{(l+1)m}$ . In this case,  $\Pr[\xi' = \xi \mid \mathcal{O}_{\text{LWE}} = \mathcal{O}_\S] = \frac{1}{2}$ . Due to the probability of executing this procedure with probability  $\frac{q_{H_1} + q_{H_2} - 1}{q_{H_1} q_{H_2}}$ , the advantage of  $\mathcal{C}$  to solve the  $\text{LWE}_{n,m,q,\chi}$  hardness is  $\frac{(q_{H_1} + q_{H_2} - 1)\epsilon_2}{2q_{H_1} q_{H_2}}$ , which is also non-negligible.

## 6. The Design of FS-BroSearch: An Enhanced Version

In this section, FS-BroSearch is proposed as an enhanced version of our BroSearch.

### 6.1. Concrete Construction

#### 6.1.1. System Initialization

In this procedure, TA initializes the entire system by calling the **Setup**( $1^\lambda$ ) algorithm. It first sets several system parameters  $n, m, q, \sigma, l$ , a gadget matrix  $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ , and a broadcast list  $\mathcal{L} = \{1, \dots, l\}$ . Secondly, TA initializes all nodes in the binary tree, and sets  $\tau$  as the depth of the binary tree, where

$T = 2^\tau$ , as in Fig. 3 (An example at  $\tau = 4$ ). Then, TA chooses several matrices  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ ,  $\mathbf{U} \in \mathbb{Z}_q^{n \times n}$  and  $\{\mathbf{A}_j^{(0)}, \mathbf{A}_j^{(1)}\}_{j \in [\tau]} \in \mathbb{Z}_q^{n \times m}$  uniformly. As same as in BroSearch, two hash functions are defined as  $H_1 : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q^{n \times m}$  and  $H_2 : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q^{n \times n}$ . Finally, TA obtains the public parameter and sends it to other entities. The public parameter is defined as:

$$pp := (n, m, q, \sigma, l, \tau, T, \mathbf{G}, \mathcal{L}, \mathbf{A}, \mathbf{U}, \{\mathbf{A}_j^{(0)}, \mathbf{A}_j^{(1)}\}_{j \in [\tau]}, H_1, H_2)$$

### 6.1.2. Key Generation

This procedure generates public and secret keys for the data sender and receiver through  $\mathbf{KeyGen}_S(pp)$  and  $\mathbf{KeyGen}_R(pp)$  algorithms, respectively.

For the data sender, this procedure is identical to that of BroSearch. TA sets the public and secret keys of the data sender as:

$$\mathbf{pk}_S := \mathbf{A}_S, \mathbf{sk}_S := \mathbf{T}_{\mathbf{A}_S}.$$

For the data receiver  $i \in \mathcal{L}$ , TA Invoke  $(\mathbf{A}_{R,i,0}, \mathbf{T}_{\mathbf{A}_{R,i,0}}) \leftarrow \text{TrapGen}(n, m, q)$  to generate an uniformly matrix  $\mathbf{A}_{R,i,0} \in \mathbb{Z}_q^{n \times m}$  and a basis  $\mathbf{T}_{\mathbf{A}_{R,i,0}} \in \mathbb{Z}^{m \times m}$  for  $\Lambda_q^\perp(\mathbf{A}_{R,i,0})$ , and defines the public and initial secret keys of the data receiver  $i$  as:

$$\mathbf{pk}_{R,i} := \mathbf{A}_{R,i,0}, \mathbf{sk}_{R,i,0} := \mathbf{T}_{\mathbf{A}_{R,i,0}}$$

At last, TA sends  $(\mathbf{pk}_S, \mathbf{sk}_S)$  and  $\{(\mathbf{pk}_{R,I,0}, \mathbf{sk}_{R,I,0})\}_{i \in \mathcal{L}}$  to a data sender and several data receivers in broadcast list  $\mathcal{L}$  over a secure channel.

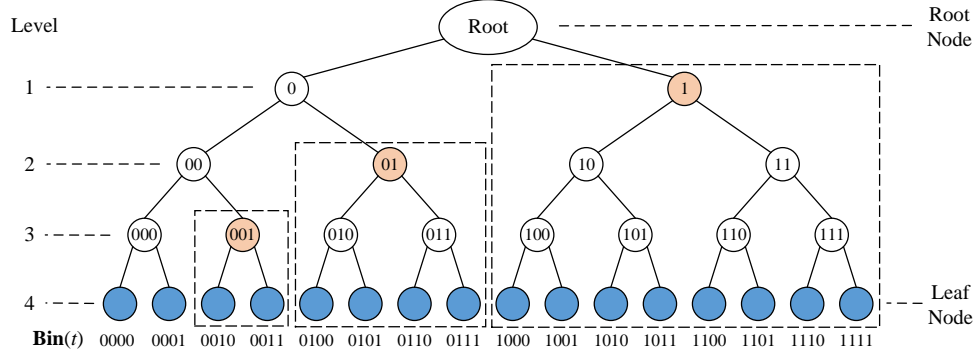


Figure 3: The binary tree utilized to secret key update for data receivers, and the number of levels  $\tau = 4$ .

### 6.1.3. Key Update

We set  $\mathbf{bin}(t)$  as the  $\tau$  bits binary representation of  $t$ ,  $\mathbf{Node}(\mathbf{bin}(t))$  as the minimal cover set of leaf node  $\mathbf{bin}(t)$ , which denotes the smallest set that includes a common ancestor node of each leaf node in  $\{\mathbf{bin}(t), \mathbf{bin}(t+1), \dots, \mathbf{bin}(T-1)\}$ , and does not include any ancestor nodes of each leaf node in  $\{\mathbf{bin}(0), \mathbf{bin}(1), \dots, \mathbf{bin}(t-1)\}$ . For example, in Fig. 3,  $\mathbf{Node}(0010) = \{001, 01, 1\}$ . The secret key of a data receiver  $i \in \mathcal{L}$  with  $t$  is defined as the set of bases corresponding to several nodes in  $\mathbf{Node}(\mathbf{bin}(t))$ .

To achieve forward security, a data receiver  $i \in \mathcal{L}$  updates its secret key through the  $\mathbf{KeyUpdate}_R(pp, \mathbf{pk}_{R,i}, \mathbf{sk}_{R,i,t})$  algorithm when the secret key is compromised. In this procedure, the data receiver inputs the secret key  $\mathbf{sk}_{R,i,t}$  with time period  $t$  and obtains  $\mathbf{sk}_{R,i,t+1}$  with  $t+1$ , where  $t \in \{0, 1, \dots, T-2\}$ . and the specific procedure can be divided into two cases.

If the time period  $t = 0$ , the data receiver owns the initial secret key  $\mathbf{sk}_{R,i,0} = \mathbf{T}_{\mathbf{A}_{R,i,0}}$  with  $t = 0$ . Due to  $\mathbf{Node}(\mathbf{bin}(1)) = \mathbf{Node}(0001) = \{0001, 001, 01, 1\}$ , it obtains  $\mathbf{sk}_{R,i,1} = \{\mathbf{T}_{\mathbf{A}_{R,i,0001}}, \mathbf{T}_{\mathbf{A}_{R,i,001}}, \mathbf{T}_{\mathbf{A}_{R,i,01}}, \mathbf{T}_{\mathbf{A}_{R,i,1}}\}$  with  $t = 1$  through calling

$$\mathbf{T}_{\mathbf{A}_{R,i,\Theta}} \leftarrow \text{ExtBasis}(\mathbf{A}_{R,i,\Theta}, \mathbf{T}_{\mathbf{A}_{R,i,0}}),$$

where  $\Theta \in \{0001, 001, 01, 1\}$ .

If the time period  $t \geq 1$ , the data receiver parses  $\mathbf{bin}(t) = (t_1, t_2, \dots, t_\tau) \in \{0, 1\}^\tau$ , and calculates the minimal cover set with time period  $t+1$ , denoted  $\mathbf{Node}(\mathbf{bin}(t+1))$ . For  $\Theta_j = (\theta_1, \dots, \theta_\zeta, \dots, \theta_j) \in \mathbf{Node}(\mathbf{bin}(t+1))$ , if there exists a basis  $\mathbf{T}_{\mathbf{A}_{R,i,\Theta_j}}$  in  $\mathbf{sk}_{R,i,t}$ , the data receiver adds this basis to  $\mathbf{sk}_{R,i,t+1}$ . Otherwise, it sets a node  $\Theta_\zeta = (\theta_1, \dots, \theta_\zeta) \in \{0, 1\}^\zeta$ , which is an ancestor node of  $\Theta_j$  in  $\mathbf{sk}_{R,i,t}$ , and then invoke the following algorithm to generate the aforementioned basis  $\mathbf{T}_{\mathbf{A}_{R,i,\Theta_j}}$ .

$$\mathbf{T}_{\mathbf{A}_{R,i,\Theta_j}} \leftarrow \text{ExtBasis}(\mathbf{A}_{R,i,\Theta_j}, \mathbf{T}_{\mathbf{A}_{R,i,\Theta_\zeta}}),$$

where  $\mathbf{A}_{R,i,\Theta_j} = (\mathbf{A}_{R,i,0} | \mathbf{A}_1^{(\theta_1)} | \dots | \mathbf{A}_j^{(\theta_j)}) \in \mathbb{Z}_q^{n \times (j+1)m}$ .

Ultimately, the data receiver  $i$  obtains  $\mathbf{sk}_{R,i,t+1} := \{\mathbf{T}_{\mathbf{A}_{R,i,\Theta_j}}\}_{\Theta_j \in \mathbf{Node}(\mathbf{bin}(t+1))}$  as the secret key with  $t+1$ .

### 6.1.4. Ciphertext Calculation

To calculate the keyword ciphertext with time period  $t$ , the data sender calls the  $\mathbf{Encrypt}(pp, \mathbf{ck}, \mathbf{pk}_S, \mathbf{sk}_S, \{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}, t)$  algorithm.

The data sender parses  $\mathbf{bin}(t) = (t_1, t_2, \dots, t_\tau) \in \{0, 1\}^\tau$ , and chooses a vector  $\mathbf{s} \in \mathbb{Z}_q^n$ , several error vectors  $\{\mathbf{e}_{R,i}\}_{i \in \mathcal{L}} \in \chi^m$ ,  $\{\mathbf{e}_{T,j}\}_{j \in [\tau]} \in \chi^m$ ,

$\mathbf{e}_K \in \chi^m, \mathbf{e}_U \in \chi^n$ . For  $j \in [\tau]$ , it calculates several vectors  $\mathbf{c}_{T,j} = (\mathbf{A}_j^{(t_j)})^\top \mathbf{s} + \mathbf{e}_{T,j} \in \mathbb{Z}_q^m$  to embed the time period into the ciphertext. As same as in BroSearch, the data sender calculates several vectors  $\mathbf{c}_{R,i} = \mathbf{A}_{R,i,0}^\top \mathbf{s} + \mathbf{e}_{R,i} \in \mathbb{Z}_q^m$  for  $i \in \mathcal{L}$ ,  $\mathbf{c}_K = H_1(\mathbf{ck})^\top \mathbf{s} + \mathbf{e}_K \in \mathbb{Z}^m$ ,  $\mathbf{c}_U = \mathbf{U}^\top \mathbf{s} + \mathbf{e}_U \in \mathbb{Z}_q^n$  and  $\boldsymbol{\varepsilon}_C \leftarrow \text{SampleLeft}(\mathbf{A}_S, \mathbf{A} + H_2(\mathbf{ck})\mathbf{G}, \mathbf{T}_{\mathbf{A}_S}, \mathbf{c}_U, \sigma)$  to achieve the ciphertext broadcast and keyword authentication.

To sum up, the data sender outputs the ciphertext corresponding to the keyword  $\mathbf{ck}$  with time period  $t$  as:

$$\text{CT}_t := (\{\mathbf{c}_{R,i}\}_{i \in \mathcal{L}}, \{\mathbf{c}_{T,j}\}_{j \in [\tau]}, \mathbf{c}_K, \boldsymbol{\varepsilon}_C).$$

#### 6.1.5. Trapdoor Generation

With an interested keyword  $\mathbf{tk} \in \mathbb{Z}_q^n$  as input, a data receiver  $\gamma \in \mathcal{L}$  executes the **Trapdoor**( $pp, \mathbf{tk}, \{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}, \mathbf{sk}_{R,\gamma,t}, \mathbf{pk}_S, t$ ) algorithm to obtain a trapdoor with the keyword  $\mathbf{tk}$  and time period  $t$ .

The data receiver first chooses a vector  $\mathbf{s}' \in \mathbb{Z}_q^n$ , several error vectors  $\mathbf{e}_S \in \chi^m, \mathbf{e}'_U \in \chi^n$ , and a matrix  $\mathbf{R}'_K \in \{-1, 1\}^{m \times m}$ . Afterward, it calculates three vectors  $\mathbf{t}_S = \mathbf{A}_S^\top \mathbf{s}' + \mathbf{e}_S \in \mathbb{Z}_q^m$ ,  $\mathbf{t}_K = (\mathbf{A} + H_2(\mathbf{tk})\mathbf{G})^\top \mathbf{s}' + (\mathbf{R}'_K)^\top \mathbf{e}_S \in \mathbb{Z}_q^m$  and  $\mathbf{t}_U = \mathbf{U}\mathbf{s}' + \mathbf{e}'_U \in \mathbb{Z}_q^n$  as the same as in BroSearch.

If  $\mathbf{sk}_{R,\gamma,t}$  does not contain  $\mathbf{T}_{\mathbf{A}_{R,\gamma}, \text{bin}(t)}$ , the data receiver invokes the following algorithms to obtain a basis  $\mathbf{T}_{\mathbf{A}_{R,\gamma}, \text{bin}(t)}$  in  $\mathbb{Z}^{(\tau+1)m \times (\tau+1)m}$  for  $\Lambda_q^\perp(\mathbf{A}_{R,\gamma,0} \mid \mathbf{A}_1^{(t_1)} \mid \dots \mid \mathbf{A}_\tau^{(t_\tau)})$ .

$$\mathbf{T}_{\mathbf{A}_{R,\gamma}, \text{bin}(t)} \leftarrow \text{ExtBasis}(\mathbf{A}_{R,\gamma,0} \mid \mathbf{A}_1^{(t_1)} \mid \dots \mid \mathbf{A}_\tau^{(t_\tau)}, \mathbf{T}_{\mathbf{A}_{R,\gamma}, \Theta_j}),$$

where  $\Theta_j$  is an ancestor node of  $\text{bin}(t)$  and  $j < \tau$ .

Following that, the data receiver generates a vector  $\boldsymbol{\varepsilon}_T$  statistically distributed in  $\mathcal{D}_{\Lambda_q^{\mathbf{t}_U}(\mathbf{A}_{R,1,0} \mid \dots \mid \mathbf{A}_{R,l,0} \mid \mathbf{A}_1^{(t_1)} \mid \dots \mid \mathbf{A}_\tau^{(t_\tau)} \mid H_1(\mathbf{tk}))}^{(l+\tau+1)m}$  through the following algorithm:

$$\begin{aligned} \boldsymbol{\varepsilon}_T \leftarrow & \text{GenSamplePre}(\mathbf{A}_{R,1,0} \mid \dots \mid \mathbf{A}_{R,l,0} \mid \mathbf{A}_1^{(t_1)} \mid \dots \mid \mathbf{A}_\tau^{(t_\tau)} \mid H_1(\mathbf{tk}), \\ & \mathbf{T}_{\mathbf{A}_{R,\gamma}, \text{bin}(t)}, \{\gamma, l+1, \dots, l+\tau\}, \mathbf{t}_U, \sigma), \end{aligned}$$

such that  $(\mathbf{A}_{R,1,0} \mid \dots \mid \mathbf{A}_{R,l,0} \mid \mathbf{A}_1^{(t_1)} \mid \dots \mid \mathbf{A}_\tau^{(t_\tau)} \mid H_1(\mathbf{tk}))\boldsymbol{\varepsilon}_T = \mathbf{t}_U \bmod q$ . Eventually, the data receiver sends the trapdoor  $\text{TD}_t := (\mathbf{t}_S, \mathbf{t}_K, \boldsymbol{\varepsilon}_T)$  to cloud server.

### 6.1.6. Search

As similar as in BroSearch, CS calculates a number  $d = \epsilon_C^\top (\mathbf{t}_S^\top \mid \mathbf{t}_K^\top)^\top - (\mathbf{c}_{R,1}^\top \mid \cdots \mid \mathbf{c}_{R,l}^\top \mid \mathbf{c}_{T,1}^\top \mid \cdots \mid \mathbf{c}_{T,\tau}^\top \mid \mathbf{c}_K^\top) \epsilon_T \in \mathbb{Z}_q$ . If  $|d| < \lfloor \frac{q}{4} \rfloor$ , this procedure outputs 1 meaning that CT and TD correspond to the same keyword, and returns the search result to the data receiver. Otherwise, this procedure outputs 0.

### 6.2. Security Analysis

**Theorem 3.** *Assuming that the  $\text{LWE}_{n,m,q,\chi}$  hardness holds, our proposed FS-BroSearch satisfies IND-CKA security in the random oracle model. For any PPT adversary  $\mathcal{A}$ , if  $\mathcal{A}$  can compromise our scheme with a non-negligible advantage  $\epsilon_3$ , then we can construct a PPT challenger  $\mathcal{C}$  to solve the  $\text{LWE}_{n,m,q,\chi}$  hardness with a non-negligible probability.*

**Proof 3.** We define an adversary  $\mathcal{A}$  and a challenger  $\mathcal{C}$  as in Theorem 1.

**LWE Instances:** This procedure is the same as Theorem 1, except that  $\{\mathbf{A}_{R,i}, \mathbf{a}_{R,i}\}_{i \in \mathcal{L}}$ ,  $(\mathbf{A}_K, \mathbf{a}_K)$  and  $\{\mathbf{A}_{T,j}, \mathbf{a}_{T,j}\}_{j \in [\tau]}$  are sampled as LWE instances in this phase.

**Init:**  $\mathcal{C}$  selects a challenge time period  $t^*$  where  $\mathbf{bin}(t^*) = (t_1^*, \dots, t_\tau^*) \in \{0, 1\}^\tau$ , a challenge public key  $\mathbf{pk}_S^* = \mathbf{A}_S$  of data sender and several challenge secret key in broadcast list  $\{\mathbf{pk}_{R,i}^* = \mathbf{A}_{R,i}\}_{i \in \mathcal{L}^*}$ .

**Setup:** Base on Theorem 1, we add the generation procedure of several matrices  $\{\mathbf{A}_i^{(0)}, \mathbf{A}_i^{(1)}\}_{i \in [\tau]}$ . Specifically,  $\mathcal{C}$  invokes  $(\mathbf{A}_0, \mathbf{T}_{\mathbf{A}_0}) \rightarrow \text{TrapGen}(n, m, q)$  to generate an uniformly matrix  $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$  and a basis  $\mathbf{T}_{\mathbf{A}_0} \in \mathbb{Z}^{m \times m}$  of  $\Lambda_q^\perp(\mathbf{A}_0)$ . For the node  $\mathcal{T}_j = (t_1, \dots, t_j)$  and  $j = [\tau]$ , if  $\mathcal{T}_j = (t_1^*, \dots, t_j^*)$ ,  $\mathcal{C}$  sets  $\mathbf{A}_j^{(t_j)} = \mathbf{A}_{T,j}$ . Otherwise,  $\mathcal{C}$  invokes  $(\mathbf{A}_j^{(t_j)}, \mathbf{T}_{\mathbf{A}_j^{(t_j)}}) \rightarrow \text{TrapGen}(n, m, q)$  to generate an uniformly matrix  $\mathbf{A}_j^{(t_j)} \in \mathbb{Z}_q^{n \times m}$  and a basis  $\mathbf{T}_{\mathbf{A}_j^{(t_j)}} \in \mathbb{Z}^{m \times m}$  of  $\Lambda_q^\perp(\mathbf{A}_j^{(t_j)})$ .

**Phase 1:**  $\mathcal{A}$  executes these following queries adaptively:

- $\mathcal{O}_{H_1}$ : This procedure is the same as Theorem 1.
- $\mathcal{O}_{H_2}$ : This procedure is the same as Theorem 1.
- $\mathcal{O}_{\text{KU}}$ : After obtaining a time period  $t$  for  $i \in \mathcal{L}$  from  $\mathcal{A}$ ,  $\mathcal{C}$  calculates the secret key  $\mathbf{sk}_{R,i,t}$  with time period  $t$ . Specifically, if  $t \geq t^*$  where  $\mathbf{bin}(t) = (t_1, \dots, t_\tau)$ ,  $\mathcal{C}$  invokes  $\mathbf{T}_{\mathcal{T}_j} \leftarrow \text{ExtBasis}(\mathbf{A}_{R,i,0} \mid \mathbf{A}_1^{(t_1)} \mid \cdots \mid$



$\mathbf{A}_j^{(t_j)}, \mathbf{T}_{\mathbf{A}_j^{(t_j)}}$ ) to generate a basis  $\mathbf{T}_{\mathcal{T}_j}$  for node  $\mathcal{T}_j$ . After that,  $\mathcal{C}$  calculates  $\mathbf{sk}_{R,i,t}$  which includes several bases in  $\text{Node}(\text{bin}(t))$  as the secret key with time period  $t$ , and returns it to  $\mathcal{A}$ .

- $\mathcal{O}_{\text{CT}}$ : After obtaining a keyword  $\mathbf{ck}$ , the receivers' public keys in broadcast list  $\{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}$  and a time period  $t$  from  $\mathcal{A}$ ,  $\mathcal{C}$  calls **Encrypt**( $pp, \mathbf{ck}, \mathbf{pk}_S^*, \mathbf{sk}_S^*, \{\mathbf{pk}_{R,i}\}_{i \in \mathcal{L}}, t$ ) algorithm to generate a ciphertext  $\text{CT}_t$ , and sends it to  $\mathcal{A}$ .
- $\mathcal{O}_{\text{TD}}$ : After obtaining a keyword  $\mathbf{tk}$ , the sender's public key  $\mathbf{pk}_S$  and a time period  $t$  from  $\mathcal{A}$ ,  $\mathcal{C}$  selects a challenge receiver  $\gamma \in \mathcal{L}^*$ , and executes the  $\mathcal{O}_{H_1}$  and  $\mathcal{O}_{H_2}$  oracle to query the hash value of  $\mathbf{tk}$ . If  $H_1(\mathbf{tk}) = \mathbf{A}_K$  or  $H_2(\mathbf{tk}) = h^*$ , this procedure is aborted by  $\mathcal{C}$ . Otherwise,  $\mathcal{C}$  calculates  $\mathbf{t}_S, \mathbf{t}_K, \mathbf{t}_U$  and  $\mathbf{T}_{\mathbf{A}_{R,\gamma,\text{bin}(t)}}$  as in **Trapdoor** algorithm, and invokes  $\epsilon_T \leftarrow \text{GenSamplePre}(\mathbf{A}_{R,1,0} \mid \cdots \mid \mathbf{A}_{R,l,0} \mid \mathbf{A}_1^{(t_1)} \mid \cdots \mid \mathbf{A}_\tau^{(t_\tau)} \mid H_1(\mathbf{tk}), \mathbf{T}_{H_1(\mathbf{tk})}, \{l + \tau + 1\}, \mathbf{t}_U, \sigma)$  such that  $(\mathbf{A}_{R,1,0} \mid \cdots \mid \mathbf{A}_{R,l,0} \mid \mathbf{A}_1^{(t_1)} \mid \cdots \mid \mathbf{A}_\tau^{(t_\tau)} \mid H_1(\mathbf{tk}))\epsilon_T = \mathbf{t}_U \bmod q$ . Finally,  $\mathcal{C}$  returns  $\text{TD}_t = (\mathbf{t}_S, \mathbf{t}_K, \epsilon_T)$  to  $\mathcal{A}$ .

**Challenge:** Base on Theorem 1, we add  $\mathbf{c}_{T,j}^* = \mathbf{a}_{T,j}$  for  $j \in [\tau]$ .

**Phase 2:** This procedure is the same as Theorem 1.

**Guess:** This procedure is the same as Theorem 1.

**Analysis:** We demonstrate two cases according to the sampling way of LWE instances. If these instances are sampled from the  $\mathcal{O}_S$ , we have:

$$\begin{aligned} \mathbf{c}_{R,i}^* &= \mathbf{a}_{R,i} = \mathbf{A}_{R,i}^\top \mathbf{s} + \mathbf{e}_{R,i}, \text{ where } i \in \mathcal{L}^*, \\ \mathbf{c}_{T,j}^* &= \mathbf{a}_{T,j} = \mathbf{A}_{T,j}^\top \mathbf{s} + \mathbf{e}_{T,j}, \text{ where } j \in [\tau] \\ \mathbf{c}_K^* &= \mathbf{a}_K = \mathbf{A}_K^\top \mathbf{s} + \mathbf{e}_K = H_1(\mathbf{ck}_\xi^*)^\top \mathbf{s} + \mathbf{e}_K, \\ \epsilon_C^* &\leftarrow \mathbb{Z}_q^{2m}. \end{aligned}$$

Thus,  $\text{CT}_{t,\xi}^* = (\{\mathbf{c}_{R,i}^*\}_{i \in \mathcal{L}^*}, \{\mathbf{c}_{T,j}^*\}_{j \in [\tau]}, \mathbf{c}_K^*, \epsilon_C^*)$  is a valid ciphertext. When  $\mathcal{A}$  compromise our scheme with advantage  $\epsilon_3$ ,  $\Pr[\xi' = \xi | \mathcal{O}_{\text{LWE}} = \mathcal{O}_S] = \frac{1}{2} + \epsilon_3$ . Otherwise,  $\text{CT}_{t,\xi}^*$  is uniform over  $(\mathbb{Z}_q^m)^l \times (\mathbb{Z}_q^m)^\tau \times \mathbb{Z}_q^m \times \mathbb{Z}_q^{2m}$ . In this case,  $\Pr[\xi' = \xi | \mathcal{O}_{\text{LWE}} = \mathcal{O}_\S] = \frac{1}{2}$ . Due to the probability of executing this procedure with probability  $\frac{q_{H_1} + q_{H_2} - 1}{q_{H_1} q_{H_2}}$ , the advantage of  $\mathcal{C}$  to solve the  $\text{LWE}_{n,m,q,\chi}$  hardness is  $\frac{(q_{H_1} + q_{H_2} - 1)\epsilon_3}{2q_{H_1} q_{H_2}}$ , which is also non-negligible.

**Theorem 4.** Assuming that the  $\text{LWE}_{n,m,q,\chi}$  hardness holds, our proposed FS-BroSearch satisfies IND-IKGA security in the random oracle model. For any PPT adversary  $\mathcal{A}$ , if  $\mathcal{A}$  can compromise our scheme with a non-negligible advantage  $\epsilon_4$ , then we can construct a PPT challenger  $\mathcal{C}$  to solve the  $\text{LWE}_{n,m,q,\chi}$  hardness with a non-negligible probability.

**Proof 4.** We define an adversary  $\mathcal{A}$  and a challenger  $\mathcal{C}$  as in Theorem 2.

**LWE Instances:** This procedure is the same as Theorem 2.

**Init:** This procedure is the same as Theorem 2.

**Setup:** This procedure is the same as Theorem 2.

**Phase 1:**  $\mathcal{A}$  executes these following queries adaptively:

- $\mathcal{O}_{H_1}$ : This procedure is the same as Theorem 2.
- $\mathcal{O}_{H_2}$ : This procedure is the same as Theorem 2.
- $\mathcal{O}_{KU}$ : This procedure is the same as Theorem 2.
- $\mathcal{O}_{CT}$ : This procedure is the same as Theorem 2, except that a time period  $t$  is obtained and  $\mathcal{C}$  calculates  $\{\mathbf{c}_{R,i}\}_{i \in \mathcal{L}}$ ,  $\{\mathbf{c}_{T,j}\}_{j \in [\tau]}$ ,  $\mathbf{c}_K$  and  $\mathbf{c}_U$  as in **Encrypt** algorithm.
- $\mathcal{O}_{TD}$ : This procedure is the same as Theorem 2, except that a time period  $t$  is obtained and  $\mathcal{C}$  calls **Trapdoor**( $pp, \mathbf{tk}, \{\mathbf{pk}_{R,i}^*\}_{i \in \mathcal{L}}, \mathbf{sk}_{R,\gamma}^*, \mathbf{pk}_S, t$ ) algorithm to generate a trapdoor  $\text{TD}_t$ .

**Challenge:** This procedure is the same as Theorem 2, except that  $\mathcal{C}$  samples  $\boldsymbol{\varepsilon}_T^* \leftarrow \mathbb{Z}_q^{(l+\tau+1)m}$ , and returns  $\text{TD}_{t,\xi}^*$  to  $\mathcal{A}$ .

**Phase 2:** This procedure is the same as Theorem 2.

**Guess:** This procedure is the same as Theorem 2.

**Analysis:** Similar to Theorem 2,  $\text{TD}_{t,\xi}^* = (\mathbf{t}_S^*, \mathbf{t}_K^*, \boldsymbol{\varepsilon}_T^*)$  is a valid ciphertext. The advantage of  $\mathcal{C}$  to solve the  $\text{LWE}_{n,m,q,\chi}$  hardness is  $\frac{(q_{H_1} + q_{H_2} - 1)\epsilon_4}{2q_{H_1}q_{H_2}}$ , which is also non-negligible.

## 7. Performance Evaluation and Comparison

We evaluate the computational and communication overheads of our BroSearch and FS-BroSearch. To ensure fairness, we set the parameters  $q = 4097$  and  $n = 64$  and compare with other lattice-based PEKS schemes [43], [15], [34], [35]. All experiments were implemented in Python using the

Table 2: Theoretical computational overhead comparison

Schemes	Encrypt	Trapdoor	Search
FS-PEKS [43]	$(nm^2 + nml_t + nl_t)T_M + T_I + T_H$	$T_{NBD} + T_{SP} + nm^2T_M + T_I + T_H$	$ml_tT_M$
PAEKS [15]	$l_sT_{SL} + (n^2m + n^2l_s + 3m^2l_s + 4nml_s)T_M + T_H$	$l_rT_{SL} + (n^2m + n^2l_r + 3m^2l_r + 4nml_r)T_M + T_H$	$8ml_sl_rT_M$
ABAEKS [34]	$T_{SL} + [(2 att  + 3)nm + n^2 + ( att  + 1)m^2]T_M + T_H$	$T_{SL} + (n^2 + m^2 + 2nm)T_M + T_{Ep_k} + T_H$	$5mT_M + T_{Ect}$
Re-PAEKS [35]	$T_{SL} + (n^2 + 3m^2 + 5nm)T_M + T_H$	$T_{SL} + (n^2 + 3m^2 + 5nm)T_M + T_H$	$8mT_M$
Our BroSearch	$T_{SL} + [n^2m + n^2 + (l + 1)nm]T_M + 2T_H$	$T_{GSP} + (n^2m + n^2 + m^2 + 2nm)T_M + 2T_H$	$(l + 3)mT_M$
Our FS-BroSearch	$T_{SL} + [n^2m + n^2 + (l + \tau + 1)nm]T_M + 2T_H$	$T_{EB} + T_{GSP} + (n^2m + n^2 + m^2 + 2nm)T_M + 2T_H$	$(l + \tau + 3)mT_M$

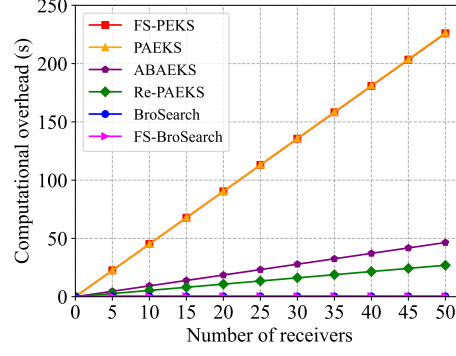
Note:  $l_t$ : The security-level of testing;  $l_s, l_r$ : The execution number of **SampleLeft** for sender and receiver defined in [15];  $|att|$ : The length of attributes;  $l$ : The number of data receivers.

NumPy library, which was conducted on a laptop with a 12th Gen Intel(R) Core(TM) i7-12800HX CPU, 16 GB of RAM, and running in Windows 10. Each experiment is conducted independently in each round.

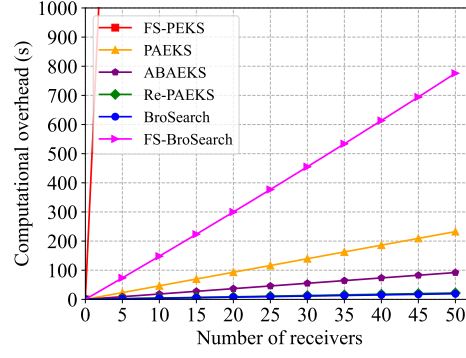
### 7.1. Computational Overhead

To conduct a theoretical comparison of the computational overhead between our two designs and other state-of-the-art approaches [43], [15], [34], [35], we focus solely on the following operations:  $T_H$  denotes the time cost of a hash function;  $T_I$  represents the time cost of matrix inversion;  $T_M$  indicates the time cost of multiplication;  $T_{NBD}$ ,  $T_{EB}$ ,  $T_{SP}$ ,  $T_{SL}$  and  $T_{GSP}$  correspond to the time costs of the **NewBasisDel**, **ExtBasis**, **SamplePre**, **SampleLeft** and **GenSamplePre** algorithms, respectively;  $T_{Ep_k}$  and  $T_{Ect}$  represent the time costs of the **Eval<sub>pk</sub>** and **Eval<sub>ct</sub>** algorithms, as defined in [34].

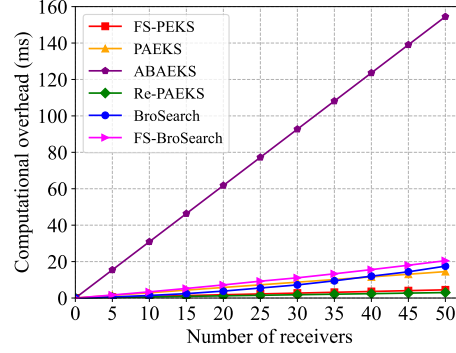
As shown in Table 2, when encrypting a keyword, our design avoids the time-consuming matrix inversion operations and multiple invocations of the **SampleLeft** algorithm, realizing much more efficient than FS-PEKS and PAEKS, and comparable to ABAEKS and Re-PAEKS. Furthermore, in a multi-receiver scenario, our scheme enables direct broadcasting of the ciphertext to the receivers, eliminating the need for additional operations such as embedding receiver attribute [34] or re-encryption [35], which is well-suited for cloud networks. For the **Trapdoor** algorithm, similar to the encryption process, our BroSearch demonstrates superior efficiency compared



(a) **Encrypt** algorithm

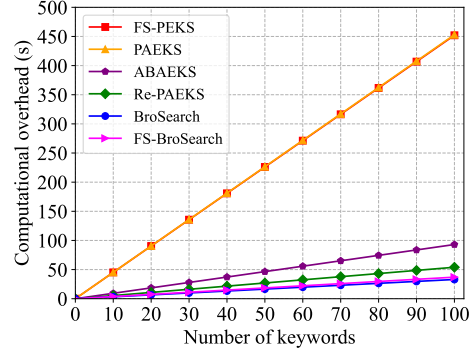


(b) **Trapdoor** algorithm

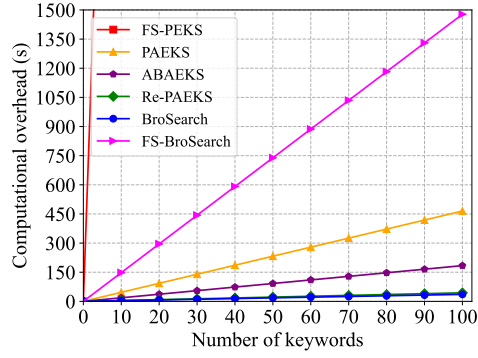


(c) **Search** algorithm

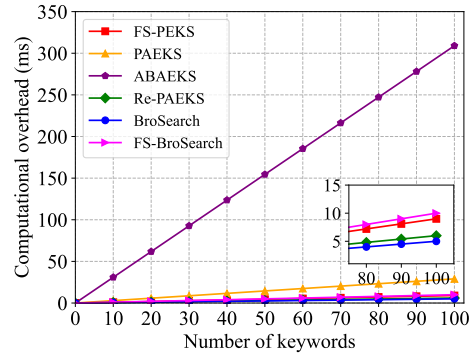
Figure 4: Computational overhead comparison with the number of data receivers  $l$ .



(a) **Encrypt** algorithm



(b) **Trapdoor** algorithm



(c) **Search** algorithm

Figure 5: Computational overhead comparison with the number of keywords  $k$ .

Table 3: The computational overhead comparison when  $k = 1$

Schemes	Encrypt	Trapdoor	Search
FS-PEKS [43]	4.52s	553.51s	0.09ms
PAEKS [15]	4.53s	4.65s	0.29ms
ABAEKS [34]	0.93s	1.84s	3.09ms
Re-PAEKS [35]	0.54s	0.44s	0.06ms
BroSearch	0.33s	0.36s	0.05ms
FS-BroSearch	0.37s	14.78s	0.10ms

to FS-PEKS and PAEKS, while remaining comparable to ABAEKS and Re-PAEKS. Further, our FS-BroSearch invokes the **ExtBasis** algorithm to achieve forward security, whereas FS-PEKS employs the **NewBasisDel** algorithm. As  $T_{NBD} \gg T_{EB}$ , the efficiency of FS-BroSearch represents a significant advantage over FS-PEKS. While this additional operation introduces computational overhead compared to PAEKS, ABAEKS, and Re-PAEKS, our FS-BroSearch offers resilience against key leakage issues. In the **Search** algorithm, as in FS-PEKS, PAEKS, and Re-PAEKS, the time cost of our schemes is primarily determined by a constant-level multiplication operation, which is more efficient than ABAEKS.

Fig. 4 demonstrates the computational overhead of our BroSearch and FS-BroSearch compared to FS-PEKS [43], PAEKS [15], ABAEKS [34], and RE-PAEKS [35] with the number of receivers  $l$ . In Fig. 4(a), since our BroSearch and FS-BroSearch allow a ciphertext to be broadcast to multiple receivers simultaneously, the time costs of the **Encrypt** algorithm in BroSearch and FS-BroSearch schemes are more efficient than the other four schemes. Fig. 4(b) illustrates the computational overhead of the trapdoor generation procedure. Due to the additional cost of invoking the **ExtBasis** algorithm, the time cost of FS-BroSearch is higher compared to BroSearch, PAEKS, ABAEKS, and Re-PAEKS, which is considered a trade-off between security and efficiency. Moreover, BroSearch demonstrates superior efficiency relative to PAEKS, ABAEKS, and Re-PAEKS. For example, when setting  $l = 1$ , PAEKS takes 4.65s, ABAEKS takes 1.84s, and Re-PAEKS takes 0.44s, while our BroSearch only takes 0.36s. In Fig. 4(c), the search cost of BroSearch and FS-BroSearch is much more efficient than that of ABAEKS and is largely consistent with FS-PEKS, PAEKS, and Re-PAEKS schemes, which are practical in cloud networks.

As depicted in Fig. 5, we illustrate the computational overhead with a different number of keywords  $k$  during ciphertext generation, trapdoor

Table 4: Theoretical communication overhead comparison

Schemes	Secret key	Ciphertext	Trapdoor
FS-PEKS [43]	$m^2 \mathbb{Z}_q $	$l_t(m+1) \mathbb{Z}_q $	$m \mathbb{Z}_q $
PAEKS [15]	$m^2 \mathbb{Z}_q $	$8ml_s \mathbb{Z}_q $	$8ml_r \mathbb{Z}_q $
ABAEKS [34]	$4m^2 \mathbb{Z}_q $	$( att +4)m \mathbb{Z}_q $	$5m \mathbb{Z}_q $
Re-PAEKS [35]	$m^2 \mathbb{Z}_q $	$8m \mathbb{Z}_q $	$8m \mathbb{Z}_q $
BroSearch	$m^2 \mathbb{Z}_q $	$(l+3)m \mathbb{Z}_q $	$(l+3)m \mathbb{Z}_q $
FS-BroSearch	$m^2 \mathbb{Z}_q $	$(l+\tau+3)m \mathbb{Z}_q $	$(l+\tau+3)m \mathbb{Z}_q $

generation, and search procedures for these six schemes. In Fig. 5(a), the time cost of our BroSearch and FS-BroSearch is significantly lower than that of FS-PEKS and PAEKS, which aligns with the theoretical values presented in Tab. 2. Moreover, since a lower-dimensional matrix is used to invoke the **SampleLeft** algorithm, the actual encryption overhead in our schemes is more efficient than that of ABAEKS and Re-PAEKS, with this advantage increasing as  $k$  grows. Fig. 5(b) and Fig. 5(c) show the computational overhead of the **Trapdoor** and **Search** algorithms in these six schemes, respectively. We observe that the time costs of our schemes are proportional to the number of keywords  $k$ , and these results are consistent with those shown in Fig. 4(b) and Fig. 4(c). For example, when  $k = 1$ , the time costs of these six schemes are shown in Table 3. Specifically, the time costs of the **Encrypt**, **Trapdoor**, and **Search** algorithms in our BroSearch are 0.33s, 0.36s, and 0.05ms, respectively, which are 61.11%, 81.82%, and 83.33% compared to the others [43], [15], [34], [35].

Notably, since the **Setup**, **KeyGen<sub>S</sub>**, and **KeyGen<sub>R</sub>** algorithms are executed less frequently than the **Encrypt**, **Trapdoor**, and **Search** algorithms in real-world applications, which have minimal impact on execution efficiency in cloud networks. Consequently, we focus only on the evaluation and comparison of ciphertext generation, trapdoor generation, and search procedures.

## 7.2. Communication Overhead

The transmission of the secret key, ciphertext, and trapdoor among participating entities contributes to the overall communication overhead. In this context, we provide a theoretical comparison of the communication overhead between our BroSearch and FS-BroSearch and other state-of-the-art schemes [43], [15], [34], [35] in Table 4, where  $|\mathbb{Z}_q|$  denotes the bit length of elements in  $\mathbb{Z}_q$ . For the data sender's secret key, the theoretical length across these six schemes is  $m^2|\mathbb{Z}_q|$ , and thus it will not be repeated. As indicated in Table 4,

the secret key size for a data receiver in our schemes is equivalent to that in FS-PEKS, PAEKS, and Re-PAEKS, and outperforms ABAEKS due to the integration of the access policy. During ciphertext and trapdoor generation, the ciphertext size in our BroSearch (and FS-BroSearch) is identical to the trapdoor size, which is influenced by the number of receivers  $l$  (and for FS-BroSearch, also by the time period  $\tau$ ). While the communication overhead of our schemes does not present a substantial advantage, our approach offers significant benefits, including ciphertext broadcasting and forward security, distinguishing it from other schemes.

Subsequently, we set the parameters as follows:  $q = 4097$ ,  $|\mathbb{Z}_q| = 13$ ,  $n = 64$ ,  $m = \lceil 2n \log q \rceil = 1537$ ,  $l_t = 10$ ,  $l_s = l_r = 4$ ,  $|att| = 10$ ,  $l = 1$ , and  $\tau = 4$ . We then evaluate the communication overhead of our BroSearch and FS-BroSearch. For example, the ciphertext size of BroSearch and FS-BroSearch is given by  $(l + 3)m|\mathbb{Z}_q| = (1 + 3) \times 1537 \times 13 \approx 9.76\text{KB}$  and  $(l + \tau + 3)m|\mathbb{Z}_q| = (1 + 4 + 3) \times 1537 \times 13 \approx 19.51\text{KB}$ , respectively. In Fig. 6, we compare these evaluation results with those of prior works [43], [15], [34], [35]. As shown, our schemes demonstrate advantages over existing methods in terms of ciphertext size.

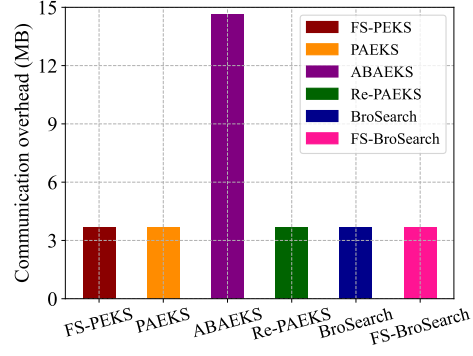
## 8. Conclusion

In this paper, we propose BroSearch, an outsourced data privacy-preserving framework through efficient broadcast encrypted search for cloud networks. Furthermore, we present a forward-secure version, called FS-BroSearch, which successfully mitigates secret key leakage problems. Rigorous security analysis demonstrates that BroSearch and FS-BroSearch achieve IND-CKA and IND-IKGA security in the ROM. Comprehensive experimental evaluations also indicate that our BroSearch offers significant advantages in computational efficiency. We acknowledge that further work is required to enhance the security level from the ROM to the standard model.

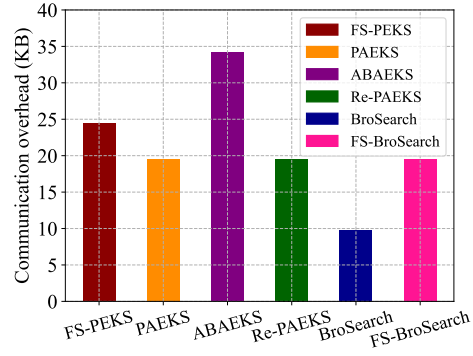
## References

- [1] Y. Yang, Y. Chen, F. Chen, J. Chen, An efficient identity-based provable data possession protocol with compressed cloud storage, *IEEE Transactions on Information Forensics and Security* 17 (2022) 1359–1371.
- [2] V. K. Tanwar, B. Raman, R. Bhargava, Crypt-or: A privacy-preserving distributed cloud computing framework for object-removal in the en-

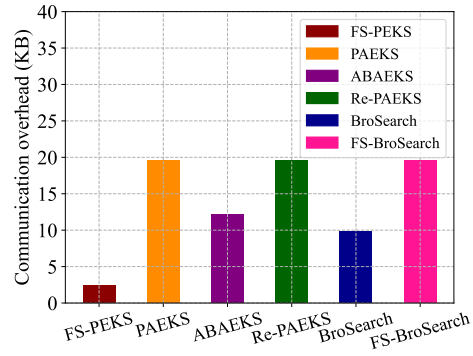




(a) Receiver's secret key size



(b) Ciphertext size



(c) Trapdoor size

Figure 6: Communication overhead comparison.

- encrypted images, *Journal of Network and Computer Applications* 208 (2022) 103514.
- [3] C. Yang, Y. Liu, Y. Ding, H. Liang, Secure data migration from fair contract signing and efficient data integrity auditing in cloud storage, *Journal of Network and Computer Applications* 239 (2025) 104173.
  - [4] N. Anjum, Z. Latif, H. Chen, Security and privacy of industrial big data: Motivation, opportunities, and challenges, *Journal of Network and Computer Applications* (2025) 104130.
  - [5] G. Xu, S. Xu, X. Fan, Y. Cao, Y. Mao, Y. Xie, X.-B. Chen, Rat ring: Event driven publish/subscribe communication protocol for iiot by report and traceable ring signature, *IEEE Transactions on Industrial Informatics* (2025) 1–9doi:10.1109/TII.2025.3567265.
  - [6] M. Ma, B. Chen, D. Tang, M. Deng, T. Xiang, D. He, Certificateless searchable public key encryption with trapdoor indistinguishability for iov, *IEEE Transactions on Vehicular Technology* 74 (3) (2025) 5085–5096.
  - [7] K. Zhang, B. Hu, J. Ning, J. Gong, H. Qian, Pattern hiding and authorized searchable encryption for data sharing in cloud storage, *IEEE Transactions on Knowledge and Data Engineering* 37 (5) (2025) 2802–2815.
  - [8] J. Li, L. Ji, Y. Zhang, Y. Lu, J. Ning, Response-hiding and volume-hiding verifiable searchable encryption with conjunctive keyword search, *IEEE Transactions on Computers* 74 (2) (2025) 455–467.
  - [9] C. Liu, L. Zhu, J. Chen, Efficient searchable symmetric encryption for storing multiple source dynamic social data on cloud, *Journal of Network and Computer Applications* 86 (2017) 3–14.
  - [10] L. Xu, J. Li, X. Chen, W. Li, S. Tang, H.-T. Wu, Tc-pedcks: Towards time controlled public key encryption with delegatable conjunctive keyword search for internet of things, *Journal of Network and Computer Applications* 128 (2019) 11–20.

- [11] Q. Huang, H. Li, An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks, *Information Sciences* 403 (2017) 1–14.
- [12] J. Tian, Y. Lu, J. Li, Lightweight searchable and equality-testable certificateless authenticated encryption for encrypted cloud data, *IEEE Transactions on Mobile Computing* 23 (8) (2024) 8431–8446.
- [13] Y. Xu, H. Cheng, J. Li, X. Liu, X. Zhang, M. Wang, Lightweight multi-user public-key authenticated encryption with keyword search, *IEEE Transactions on Information Forensics and Security* 20 (2025) 3234–3246.
- [14] Z.-Y. Liu, Y.-F. Tseng, R. Tso, M. Mambo, Y.-C. Chen, Public-key authenticated encryption with keyword search: A generic construction and its quantum-resistant instantiation, *The Computer Journal* 65 (10) (2022) 2828–2844.
- [15] L. Cheng, F. Meng, Public key authenticated encryption with keyword search from lwe, in: *European Symposium on Research in Computer Security*, Springer, 2022, pp. 303–324.
- [16] S. Xu, Y. Cao, X. Chen, Y. Zhao, S.-M. Yiu, Post-quantum public-key authenticated searchable encryption with forward security: General construction, and applications, in: *International Conference on Information Security and Cryptology*, Springer, 2023, pp. 274–298.
- [17] L. Yao, J. Weng, A. Yang, X. Liang, Z. Wu, Z. Jiang, L. Hou, Scalable cca-secure public-key authenticated encryption with keyword search from ideal lattices in cloud computing, *Information Sciences* 624 (2023) 777–795.
- [18] L. Cheng, F. Meng, Server-aided public key authenticated searchable encryption with constant ciphertext and constant trapdoor, *IEEE Transactions on Information Forensics and Security* 19 (2024) 1388–1400.
- [19] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, B. Wang, Efficient encrypted keyword search for multi-user data sharing, in: *Computer Security—ESORICS 2016: 21st European Symposium on Research in Computer Security*, Heraklion, Greece, September 26–30, 2016, Proceedings, Part I 21, Springer, 2016, pp. 173–195.

- [20] B. Zhang, W. Yang, F. Zhang, J. Ning, Efficient attribute-based searchable encryption with policy hiding over personal health records, *IEEE Transactions on Dependable and Secure Computing* 22 (2) (2025) 1299–1312.
- [21] G. Xu, X. Fan, S. Xu, Y. Cao, X.-B. Chen, T. Shang, S. Yu, Anonymity-enhanced sequential multi-signer ring signature for secure medical data sharing in iomt, *IEEE Transactions on Information Forensics and Security* 20 (2025) 5647–5662.
- [22] X. Liu, K. He, G. Yang, W. Susilo, J. Tonien, Q. Huang, Broadcast authenticated encryption with keyword search, in: *Australasian Conference on Information Security and Privacy*, Springer, 2021, pp. 193–213.
- [23] S. Mukherjee, Statistically consistent broadcast authenticated encryption with keyword search: Adaptive security from standard assumptions, in: *Australasian Conference on Information Security and Privacy*, Springer, 2023, pp. 523–552.
- [24] K. Emura, et al., Generic construction of fully anonymous broadcast authenticated encryption with keyword search with adaptive corruptions, *IET Information Security* 2023 (2023).
- [25] G. Xu, S. Xu, Y. Cao, K. Xiao, Y. Mao, X.-B. Chen, M. Dong, S. Yu, Aaq-peks: An attribute-based anti-quantum public key encryption scheme with keyword search for e-healthcare scenarios, *Peer-to-Peer Networking and Applications* 18 (2) (2025) 1–21.
- [26] X. Chen, S. Xu, Y. Cao, Y. He, K. Xiao, Aqrs: Anti-quantum ring signature scheme for secure epidemic control with blockchain, *Computer Networks* 224 (2023) 109595.
- [27] J. Jiang, D. Wang, Qpase: Quantum-resistant password-authenticated searchable encryption for cloud storage, *IEEE Transactions on Information Forensics and Security* 19 (2024) 4231–4246.
- [28] S. Xu, Y. Cao, X. Chen, Y. Guo, Y. Yang, F. Guo, S.-M. Yiu, Post-quantum searchable encryption supporting user-authorization for outsourced data management, in: *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 2702–2711.

- [29] Z. Lin, H. Li, X. Chen, M. Xiao, Q. Huang, Identity-based encryption with disjunctive, conjunctive and range keyword search from lattices, *IEEE Transactions on Information Forensics and Security* 19 (2024) 8644–8657.
- [30] Y. Cao, S. Xu, X. Chen, Y. He, S. Jiang, A forward-secure and efficient authentication protocol through lattice-based group signature in vanets scenarios, *Computer Networks* 214 (2022) 109149.
- [31] X. Yu, L. Xu, X. Huang, C. Xu, An efficient lattice-based encrypted search scheme with forward security, in: *International Conference on Network and System Security*, Springer, 2022, pp. 712–726.
- [32] X. Chen, S. Xu, S. Gao, Y. Guo, S.-M. Yiu, B. Xiao, Fs-llrs: Lattice-based linkable ring signature with forward security for cloud-assisted electronic medical records, *IEEE Transactions on Information Forensics and Security* 19 (2024) 8875–8891.
- [33] Y. Ge, Y. Gao, J. Ning, J. Ma, X. Chen, Verifiable multilevel dynamic searchable encryption with forward and backward privacy in cloud-assisted iot, *IEEE Internet of Things Journal* 11 (24) (2024) 40861–40874.
- [34] F. Luo, H. Wang, C. Lin, X. Yan, Abaeks: Attribute-based authenticated encryption with keyword search over outsourced encrypted data, *IEEE Transactions on Information Forensics and Security* 18 (2023) 4970–4983.
- [35] F. Luo, H. Wang, X. Yan, Re-paeks: Public-key authenticated re-encryption with keyword search, *IEEE Transactions on Mobile Computing* 23 (10) (2024) 10077–10092.
- [36] L. Chen, J. Li, J. Li, J. Weng, Paess: Public-key authentication encryption with similar data search for pay-per-query, *IEEE Transactions on Information Forensics and Security* 19 (2024) 9910–9923.
- [37] Z.-Y. Liu, Y.-F. Tseng, R. Tso, M. Mambo, Y.-C. Chen, Public-key authenticated encryption with keyword search: Cryptanalysis, enhanced security, and quantum-resistant instantiation, in: *Proceedings of the 2022 ACM on Asia conference on computer and communications security*, 2022, pp. 423–436.

- [38] A. Fiat, M. Naor, Broadcast encryption, in: *Advances in Cryptology—CRYPTO’93: 13th Annual International Cryptology Conference* Santa Barbara, California, USA August 22–26, 1993 Proceedings 13, Springer, 1994, pp. 480–491.
- [39] M. A. R. Baee, L. Simpson, X. Boyen, E. Foo, J. Pieprzyk, Ali: Anonymous lightweight inter-vehicle broadcast authentication with encryption, *IEEE Transactions on Dependable and Secure Computing* 20 (3) (2023) 1799–1817.
- [40] J. Zhang, S. Su, H. Zhong, J. Cui, D. He, Identity-based broadcast proxy re-encryption for flexible data sharing in vanets, *IEEE Transactions on Information Forensics and Security* 18 (2023) 4830–4842.
- [41] H. Yin, Y. Zhu, G. Guo, W. C.-C. Chu, Privacy-preserving smart contracts for confidential transactions using dual-mode broadcast encryption, *IEEE Transactions on Reliability* 73 (2) (2023) 1090–1103.
- [42] M. Ali, H. Ali, T. Zhong, F. Li, Z. Qin, A. A. Ahmed Abdelrahman, Broadcast searchable keyword encryption, in: *2014 IEEE 17th International Conference on Computational Science and Engineering*, 2014, pp. 1010–1016.
- [43] X. Zhang, C. Xu, H. Wang, Y. Zhang, S. Wang, Fs-peks: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial internet of things, *IEEE Transactions on Dependable and Secure Computing* 18 (3) (2021) 1019–1032.
- [44] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, *Journal of the ACM (JACM)* 56 (6) (2009) 1–40.
- [45] C. Gentry, C. Peikert, V. Vaikuntanathan, Trapdoors for hard lattices and new cryptographic constructions, in: *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008, pp. 197–206.
- [46] S. Agrawal, D. Boneh, X. Boyen, Efficient lattice (h) ibe in the standard model, in: *Advances in Cryptology—EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, French Riviera, May 30–June 3, 2010. Proceedings 29, Springer, 2010, pp. 553–572.

- [47] D. Cash, D. Hofheinz, E. Kiltz, C. Peikert, Bonsai trees, or how to delegate a lattice basis, *Journal of Cryptology* 25 (2012) 601–639.
- [48] D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, D. Vinayagamurthy, Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits, in: *EUROCRYPT*, Springer, 2014, pp. 533–556.